BrightStor™ CA-Allocate™ DASD Space and Placement

User Guide

5.3 SP 06

Computer Associates™

This documentation and related computer software program (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by Computer Associates International, Inc. ("CA") at any time.

This documentation may not be copied, transferred, reproduced, disclosed or duplicated, in whole or in part, without the prior written consent of CA. This documentation is proprietary information of CA and protected by the copyright laws of the United States and international treaties.

Notwithstanding the foregoing, licensed users may print a reasonable number of copies of this documentation for their own internal use, provided that all CA copyright notices and legends are affixed to each reproduced copy. Only authorized employees, consultants, or agents of the user who are bound by the confidentiality provisions of the license for the software are permitted to have access to such copies.

This right to print copies is limited to the period during which the license for the product remains in full force and effect. Should the license terminate for any reason, it shall be the user's responsibility to return to CA the reproduced copies or to certify to CA that same have been destroyed.

To the extent permitted by applicable law, CA provides this documentation "as is" without warranty of any kind, including without limitation, any implied warranties of merchantability, fitness for a particular purpose or noninfringement. In no event will CA be liable to the end user or any third party for any loss or damage, direct or indirect, from the use of this documentation, including without limitation, lost profits, business interruption, goodwill, or lost data, even if CA is expressly advised of such loss or damage.

The use of any product referenced in this documentation and this documentation is governed by the end user's applicable license agreement.

The manufacturer of this documentation is Computer Associates International, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227-7013(c)(1)(ii) or applicable successor provisions.

© 2002 Computer Associates International, Inc. (CA)

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Contents

Chapter 1: Introduction

Chapter 2: Concepts and Facilities

Chapter 3: Installing

Chapter 4: Implementation

Chapter 5: Operation

Chapter 6: User Exits

Chapter 7: Quota

Appendix A: Diagnostics

Appendix B: Bypassing Allocations

Appendix C: Dormant Operation

Appendix D: Message Control Facility

Appendix E: Optional Maintenance

Appendix F: Reports

Appendix G: ISPF Interface

Index

Chapter 1

Introduction

This guide provides instructions for the operation of BrightStor CA-Allocate DASD Space and Placement, Release 5.3. To find out more about BrightStor CA-Allocate see the *Getting Started* and the *Messages Guide.*

System Overview

BrightStor CA-Allocate is a system utility that allows the Storage Administrator to control the allocation of MVS data sets. It gets control for all requests for new disk space from both batch and online functions. Decisions can be made to either enforce standards or to allow selected jobs or users to bypass them.

As a system utility program for the MVS operating system, it controls how storage space is allocated. It intercepts requests for storage space and executes an Allocation Selection Routine (ASR) that is written in a CLIST-like language. It also provides many built-in variables, such as DSN and USER, on which the Storage Administrator can base decisions for controlling allocations.

BrightStor CA-Allocate resides in the operating system and is initiated by a Started Task. The Started Task dynamically reconfigures options through the operator's console. If the Quota selectable unit is not being used, the Started Task can be instructed to terminate immediately after installation. The system can easily be removed by an operator command to the Started Task.

It provides direct interfaces to the ACF2, eTrust CA-Top Secret, and RACF security systems. It interfaces with the installed security system to determine user IDs and to make sure that allocations for certain data sets are allowed onto certain volumes.

It can also be used to enforce an installation's naming standards. This can be done by either defining what names (or name patterns) are allowed or, optionally, which users are authorized to use each data set name.

It can control the assignment of retention periods and expiration dates during the allocation process.

The Storage Administrator can set minimum, maximum, and default retention periods at the data set, pool, or system-wide level.

It provides additional features if data set usage is tracked by groups rather than by individual users. It can define groups explicitly or, if using its RACF security interface, it can obtain the group ID the user is associated with in RACF. This use of groups minimizes the repetition of user IDs when defining installation options.

The Allocation Selection Routine (ASR) can call up to ten User Exits at any point in the routine. If special requirements exist at an installation, these exits can be coded to further tailor BrightStor CA-Allocate. It is, however, designed with maximum flexibility as its goal. Most installations are able to use the product to provide the needed functions without having to code any special routines of their own.

BrightStor CA-Allocate provides features that can dynamically enforce standards and DASD quota limits and provide pooling capabilities for better DASD utilization. With this product, the storage administrator can:

- Change the unit name.
- Optimize block size.
- Enforce cataloging action.
- Ignore or change volser parameters.
- Enforce data set naming standards.
- Execute an assembler language user exit.
- Add or remove the RLSE or CONTIG JCL parameters.
- Restrict VSAM data sets to certain catalogs.
- Prevent the allocation of non-conforming data sets.
- Add, remove, or change a retention period or expiration date.
- Provide dynamic storage group reconfiguration without an IPL.
- Change both the primary and secondary amount of space requested.
- Define which volumes are contained in each storage group (pool).
- Send a tailored message to the user making the allocation request.
- Check authority to create a data set on a volume via the security system.
- Provide an option to maintain volume or storage group free-space thresholds.
- Eliminate concern about mount attributes because volumes can be mounted with any attribute.
- Establish, maintain, and enforce disk space limits using the Quota selectable unit.

- Add, remove, or change the SMS constructs: DATACLAS, STORCLAS, MGMTCLAS, and STORGRP.
- For VSAM allocations, ignore JOBCAT and STEPCAT DD statements or a catalog name specified on the define.
- Allocate the DATA component and INDEX component of a VSAM cluster on separate volumes for better performance.
- Prevent B37- and E37-type abends by dynamically adding volumes to output data sets.

Product Overview

BrightStor CA-Allocate consists of three major components:

1. An interface to data set allocation and catalog management.
2. A utility for installing or removing the interfaces and changing processing options dynamically.
3. Product bridges to BrightStor CA-Vantage's Allocation Manager.

How It Works

BrightStor CA-Allocate intercepts data set allocation requests. After intercepted, these allocation requests can be modified.

What attributes can be modified varies, depending on the type of data set and the kind of allocation request. New data sets intercepted before their allocation requests complete have the most modifiable attributes, whereas old data sets intercepted before being renamed or scratched have the least.

Detailed information on where it intercepts data set allocation requests can be found by reviewing **TABLE 2-2 Where the Code** Gets Control. Detailed information on which data set attributes can be modified and when they can be found by reviewing **TABLE 4-3 List of Variables**.

SVC and DADSM Branch Entry Interface

BrightStor CA-Allocate intercepts all accesses to DADSM allocation routines. When installed, it replaces the address to DADSM and SVC 32 with its own DADSM routine address. Its interface to DADSM informs allocation routines when it is performing a modified allocation. After the DADSM interface is complete, normal DADSM processing continues.

SMS Subsystem Interface

In order for BrightStor CA-Allocate to redirect allocations during DFSMShsm RECALLs and RECOVERs, which are **not** SMS-managed, the IBM SMS (System Managed Storage) subsystem must be installed and active on your system. You do not have to configure IBM's SMS to do anything in particular, the subsystem just needs to be installed and active.

System Requirements and Restrictions

- It runs on all supported levels of OS/390 and z/OS.
- Allocation Manager Selectable Unit requires approximately 60KB of CSA below and 880KB above the 16MB line.
- Its Allocation Selection Routines (ASRs) are loaded into CSA above the 16MB line. The size of the ASRs varies depending on the number of statements coded.
- Because of the method used to install its allocation interface, any IPL of the system removes it from the operating system. Therefore, after a site has completed testing this product and is satisfied with its results, the site should modify its IPL procedures to include the execution of the Started Task.
- VVDS data sets should be explicitly excluded from processing. This can be accomplished by adding the following statement to the beginning of the ASR:

  ```
  IF &DSN = SYS1.VVDS.** THEN EXIT CODE(0)
  ```

- If an IDCAMS DEFINE contains the MODEL or UNIQUE parameters, or an IDCAMS IMPORT contains the VOLUME parameter, IDCAMS attempts to mount the volume specified in the VOLUME parameter. If the volume is not valid (it is an old volser or a pool name), a mount message is issued to the operator console.

  When the operator replies "no" to the volume mount request, the define is failed. All this takes place within IDCAMS, before IDCAMS issues the Catalog Management request (SVC 26). BrightStor CA-Allocate interface is in front of Catalog Management, not in IDCAMS. Therefore, if the above situation occurs, the code is never entered and therefore cannot prevent the mount from occurring.
- The VSAM component currently intercepts only DEFINEs for Clusters and Alternate Indexes. All other Catalog Management requests (ALTER, DELETE, DEFINE PATH, and so on) are unaffected.
- The CSA Quota Table used by the Quota selectable unit is allocated above the 16MB line. Its size is dependent on the number of Quota Groups defined.

- The Quota selectable unit requires that the Started Task be kept running at all times. This is necessary for the QSYNC process, described in the section **The QSYNC Process** in the chapter "**Quota**."
- It is designed to operate in a JES2 environment. While most functions operate in a JES3 environment, certain functions do not, and can cause unexpected results.
- Since JES3 does not allow dynamic allocation to tape, BrightStor CA-Allocate cannot redirect disk allocations to tape.

Special Considerations

This section covers special considerations that should be observed when using this product with other program products.

ACS Environment

The ACS environment is the earliest interface point in the allocation process. If messages are written by BrightStor CA-Allocate during JCL allocations, they are displayed before normal job startup messages. Because it is too early to scan for DDs in the ACS environment at batch JCL allocation time, DDs cannot be used as switches.

For example, VDSBYPAS, VDSDIAGS, and #VAMTST# DDs are not detectable or usable.

If the ACS environment must be bypassed for Batch JCL allocations, code the following in the ASR:

```
IF &VAMENVIR = 'ACS' THEN EXIT CODE(0)
```

To get diagnostics, BrightStor CA-Allocate must be running in ACTIVE mode and the DIAGS= option must be used. For details, see the sections **DIAGS=** Subcomand and DIAGS= Subcommand in the chapter "**Operation**."

Cobol II Users

As delivered, BrightStor CA-Allocate's non-VSAM End-Of-Volume Support excludes data sets that have either an ABEND or EOV exit coded in their DCB's exit parameter list. Some Cobol II allocations are considered to be ineligible for EOV processing for this reason. One of the following Optional Maintenance features can be used to disable such exclusions:

- **601765**
- **601769**

For more information, see the chapter "**Optional Maintenance**."

DB2

The VSAM support in the EXTEND environment does not intercept requests associated with DB2 LSDS clusters, or KSDS clusters with either the KEYRANGE or IMBED attribute.

DFDSS

BrightStor CA-Allocate is unable to successfully redirect non-SMS-managed DFDSS-requested allocations. Unpredictable results can occur if the bypass DD statement, default name VDSBYPAS, is not used to exempt such allocations from processing.

Enqueue Propagation for the DQT

The Disk Quota Table (DQT) is periodically synchronized with the CSA Quota Table (CQT) to ensure that both contain the same information. This synchronization occurs during the QSYNC process.

During the QSYNC, a SYSTEMS enqueue serializes access to the quota tables across all systems using qname STERLING. To avoid corruption, do not make changes to GRS or MIM that might prevent cross system enqueues using the STERLING qname. In particular, do not change the scope from SYSTEMS to SYSTEM.

Enqueue for the CQT

The historical exclusive ENQ on the CSA QUOTA TABLE (CQT) has been changed to a conditional one in order to eliminate the possibility of an ENQ deadlock condition. This may result in QUOTA processing being bypassed if the ENQ on the CQT can not be obtained after a maximum time duration of two (2) seconds.

FTP

FTP allocation will release unused space by default. BrightStor CA-Allocate can be used to prevent this in one of two ways:

1. EXIT CODE(8) in the RELEASE Environment
2. SET &RLSE = 'N' in the ALLOC Environment.

IAM

BrightStor CA-Allocate can successfully redirect IAM allocations after:

1. The IAM Global Change Facility has been configured to specify 'ENABLE=VAM.'
2. IAM should be started prior to starting CA-Allocate.

Note: Also check the section in the IAM manual dealing with activating the IAM VSAM Interface (VIF) and the section "Coexistence with Other Software Products."

PLSOPT6 (Y) must be specified in the PARMLIB in order to activate the new EOV Support for IAM file update operations, which is only applicable to releases V6.4/07P and above of IAM. Activating this support for earlier releases of IAM will result in C03 abends when those data sets are closed.

IDCAMS BLDINDEX Workfiles

BrightStor CA-Allocate cannot modify the unit and volume coded for the IDCUT1 and IDCUT2 work files. You must code the actual UNIT and VOL=SER parameters.

Initializing Tapes

When executing IEHINITT to initialize tapes use a VDSBYPAS DD statement in the job or add code in the VDSPROG to bypass these jobs. If &PGM='IEHINITT' THEN EXIT CODE(0). It is also recommended to add a statement in the VDSPROG to exclude any other program that initializes the tapes.

ISAM

BrightStor CA-Allocate cannot recover ISAM data sets from end-of-volume conditions. Defining ISAM data sets as multi volume ones may prevent end-of-volume conditions.

MVS System Logger Data Sets

System Logger data sets can be either SMS-managed or NonSMS. Both are VSAM data sets allocated via SVC99. To make them SMS-managed, redirect them in the ACS Environment to a DFSMS Storage Group. To make them non-SMS-managed, redirect them in the ALLOC Environment to a BrightStor CA-Allocate NonSMS Storage Group.

BrightStor CA-Compress Data Compression

Use the SHRINKTASK variable to verify that BrightStor CA-Compress Data Compression (BrightStor CA-Compress) is active on the system before BrightStor CA-Allocate redirects the restore of a BrightStor CA-Compress data set. If BrightStor CA-Compress is not active, BrightStor CA-Allocate should deny such restore requests. Failure to do so results in the loss of the BrightStor CA-Compress Safeguard Information.

BrightStor CA-Disk Backup and Restore, ROSCOE, and WIZARD

These products are known to use internal parameters to control JCL step termination disposition. These parameters are inaccessible. Unpredictable results occur if BrightStor CA-Allocate is used to change JCL step termination disposition (&DISPN) for non-VSAM data set allocations requested by any of these products.

For data sets cataloged for autorestore (DMSAR), the variable &IFALREADYCAT specifies a 'Y'.

PLSOPT16 (Y) is required in order for BrightStor CA-Disk Backup and Restore (BrightStor CA-Disk) operations to be considered eligible for BrightStor CA-Allocate’s EOV Support.

SMS-Managed GDGs

The pair of major / minor enqueues associated with DFSMS's automatic scratch of the oldest generation of an SMS-managed GDG (following the successful creation of a new generation of that same GDG) are identical to but gotten in precisely the opposite order as the pair of major / minor enqueues IBM code associates with the SVC26 request(s) BrightStor CA-Allocate issues when it is retrieving the IFCAT variables. To avoid a possible deadly embrace, BrightStor CA-Allocate has stopped issuing IFCAT-related SVC26 requests for SMS-managed GDGs. This often results in an IFCAT value of "?". Additionally, because those automatic scratches are actually a function of DFSMS's GDG ROLLIN process, and as such need to be allowed to complete successfully, they are now considered ineligible for SCRATCH Environment processing.

SnapShot

Exclude SnapShot target data sets from BrightStor CA-Allocate EOV, EOV_VSAM, and VSAM Extend environments.

According to IBM, the SIBBATCH program cannot handle any space change or volume redirection done by any other OEM product at this time.

To exclude in the EOV and EOV_VSAM environments:

```
IF &VAMENVIR = 'EOV_VSAM' AND &PGM = 'SIBBATCH' THEN EXIT CODE(0)
IF &VAMENVIR = 'EOV' AND &PGM = 'SIBBATCH' THEN EXIT CODE(0)
```

For VSAM EXTEND environment, apply the GA PTF for PI 369129 (SS04708).

MainView SRM (formerly STOP-x37)

This program product has the capability of changing the size of a data set's secondary extent.

Unpredictable results occur if both products attempt to do this for the same allocation.

XCOM

BrightStor CA-Allocate can successfully redirect XCOM allocations after the P74730 XCOM R2.3.0 ZAP has been applied.

XPEDITER

For XPEDITER allocations, the variable &IFALREADYCAT specifies a 'Y'.

How to Use this Guide

We advise new users to read the following chapters of this manual before getting started:

- Introduction
- **Concepts and Facilities**
- **Implementation**
- **Operation**

Reviewing these chapters prepares you for installation as described in the chapter "**Installing**."

Document Notation Conventions

In the body of the text, examples of JCL statements, program statements, and operator console commands are shown in a smaller print style in ALL CAPITALS and indented from the proceeding text, as shown in the example below:

```
WRITE 'DSN = &DSN, DISP = &DISP'
```

The following notation conventions are used to describe ASR statements:

- Terms shown in ALL CAPITAL letters should be typed into the ASR exactly as shown.
- Terms shown in lower case letters should be replaced or filled in as required for the given statement.
- Terms contained within brackets [ ] are optional.

An example of an ASR statement is given below:

```
IF expression THEN statement [ELSE statement]
```

Within tables in Chapters "Implementation", "Operation", and "Quota", the following notation conventions are used to describe the syntax of the Quota Configuration control statements and the operator commands.

TABLE 1-1 Notation Conventions used within Tables

NOTATION	CHARACTER	DESCRIPTION
Brackets	[ ]	Indicates optional elements that you may or may not code.
Braces	{ }	Indicates alternative elements from which you must choose one, and only one, element.
Or sign	|	Indicates that no more than one of the items that are separated with this character may be selected.
Ellipses	...	Indicates that the elements can be repeated.
Parentheses Commas Equal Signs	( ) , =	Represents characters that should be enter exactly as shown.
Bold type	**PARM**	Indicates elements that you must code exactly as shown. These elements consist of statement names, keywords, and other punctuation symbols (for example, commands, parentheses, and equal signs).
Underscored and bold	**PARM**	Indicates alternative choices that are assumed if you do not code the optional elements (that is, default values).
Italics	*PARM*	Indicates fields you supply.

Customer Support Services

Computer Associates takes pride in creating leading edge products using state of the art software technology. Of equal importance is providing high quality technical support. The guidelines below help us give you better, faster service. Refer any comments about software or service to Customer Support. We always welcome suggestions to provide better service.

Contacting CA Customer Support

Before You Call

You can help us serve you better by performing the following simple steps before you call Customer Support:

- Read the documentation carefully.
- Reconstruct the events and write them down.

Where to Call

Telephone: (800) 889-0226

Fax: (916) 463-8870

Support Representatives

Unless you are already working with someone on your particular problem, do not ask for a specific support representative.

Incomplete Information

Incomplete information can delay or prevent a solution to your problem. If you cannot provide the information, just let us know.

Fix Confirmation

Confirm the results of the fix to the support representative after you receive and test it. Let us know if it solved the problem.

System Dumps

To troubleshoot problems, a full SYSMDUMP or an SVC dump is usually needed. An ABEND-AID dump rarely gives enough information. If your site uses ABEND-AID, find out how to get a full SYSMDUMP. To generate a SYSMDUMP, supply a //SYSMDUMP DD statement defining a 3480 cartridge.

What to do When You Call In

Tell Us About Your Organization and Yourself

Your customer ID or organization name and your name

Your Environment

- The release that you are running
- The operating system and release you are running (for example, MVS 5.2 with DFSMS 1.3)
- Any changes made to your system between the time the system worked and when it didn't work.

The Problem

- Has the problem occurred before?
- Can you recreate the problem?
- The exact sequence of events that led to the error.

Collect the Documents that Describe the Problem

- The JCL, sysouts, and console log from all jobs involved.
- The message numbers and complete text of any messages that were issued.
- The message numbers and complete text of any system messages that were issued (for example, from the JES2 log).
- The abend code, module, and offset, if there was an abend.
- Your compiled ASRs, which are found in the DD SYSPRT of the Allocate STC or in a data set, depending upon the setting of VKGPARM PLSXRFDS.
- If the problem can be easily reproduced, turn on full diagnostics by issuing the command F VAM,DIAGS=jobname.

What Happens to Reported Problems?

After a problem is reported to Customer Support, it is added to StarTrak, where it becomes available for viewing and updating by the customer using StarTCC. When a resolution to the problem is found—identified as a Solution—it is tested and certified before it is made available to customers. Of course, customers who have reported the problem receive the Solution as soon as it becomes available. Confirmed Solutions are then made available on http://webtrack.ca.com for our customers. Included with each Solution is a detailed description of the problem and a description of the fix.

Chapter

2

Concepts and Facilities

In this section, we present some guidelines to consider during implementation. It is not intended as an exhaustive study. Rather, it is presented as an aid in understanding the facilities that this product offers. Each of the concepts is presented under its own heading to assist you in finding a specific area of interest. The following topics are included:

Why Use this Product?

BrightStor CA-Allocate performs several important functions:

- Simplifies the task of allocating data sets for end users.
- Enables you to use the system to enforce installation standards.
- Enables you to establish DASD space limits for individuals or groups of users. The system automatically monitors the limits and optionally enforces them.

Depending on the function, the Storage Administrator can determine the appropriate level of enforcement. This might be to simply issue a warning message and allow the allocation to continue, or possibly to cancel the allocation whenever a violation takes place. BrightStor CA-Allocate can be configured to correct the violation automatically whenever it can and to continue with the allocation process. It is entirely up to the Storage Administrator to decide just how much control it is to have.

Without BrightStor CA-Allocate, the data center must rely on its users to put their data sets on the right volumes. As more and more volumes are brought online, data set placement becomes unmanageable. Volume pooling simplifies this process by reducing the number of items that need to be managed. Instead of managing 1,000 volumes, the Storage Administrator can choose to manage 10 pools or, in BrightStor CA-Allocate terms, storage groups.

From a Storage Administrator's point of view, BrightStor CA-Allocate is the tool needed to enforce the standards and DASD space limits that users too often “forget.” It can enforce standards and DASD space limits automatically, freeing the Storage Administrator from having to correct violations after the fact.

Basics

BrightStor CA-Allocate allows the Storage Administrator to logically associate one or more volumes to form a common grouping called a storage group. The volumes typically share common characteristics. For example, they might all be 3380 devices used for production jobs by the accounting department.

After these logical pools have been defined and implemented, users can remove the JCL VOLUME parameter.

Before BrightStor CA-Allocate can control allocation of data sets, several parameter files must be created and stored as members of a partitioned data set resembling in function SYS1.PARMLIB. Each member is listed in the following table. For more information about these members, see the chapters "**Implementation**" and "**Quota**."

Note: The QREBUILD, QSCAN and QCONFIG members are only needed if the Quota selectable unit is present.

TABLE 2-1 Parameter Files

MEMBER	DESCRIPTION
CONFIG	Used during Allocate startup to a point to desired member for VKGPARMS.
QCONFIG	Quota Configuration File that defines the Quota processing options and optional manually defined Quota Groups.
QREBUILD	Quota Volume Selection ASR that is executed for every volume that is online at the time the Quota Table is rebuilt. This is for the QREBUILD environment.
QSCAN	Quota VTOC Scan ASR that is executed for every data set in each Volume Table of Contents (VTOC). This is for the QSCAN environment.
VDSPROG	Allocation Selection Routine (ASR) that is executed for every new allocation request. This member contains the logic for ACS, ALLOC, DEFINE, EOV, EOV_VSAM, EXTEND, OLD, QUOTA, RELEASE, RENAME, SCRATCH, and SPACE environments.
VDSTORGP	Storage Group definitions that assign names to groups of disk volumes (storage pools).
VKGPARMS	This member allows the Storage Administrator to customize the data set names, member names, special DD names, and alter other operational parameters.

After these members are created, the MVS operator can start BrightStor CA-Allocate by entering the following command:

```
S VAM
```

During startup, it does the following:

1. Reads system parameters.
2. Compiles the ASRs.
3. Reads and validates the Storage Group Definitions.
4. Reads and validates the Quota Configuration File.
5. If all parameters processed in steps 1 through 4 are satisfactory, then BrightStor CA-Allocate loads its components into CSA and SQA memory and intercepts:
   - DADSM SVC (SVC 32).
   - The DADSM pre-allocation and post-allocation exits (IGGPRE00 and IGGPOST0).
   - Allocation (SVC 99).
   - Catalog Management SVC (SVC 26).
   - End-of-Volume DASD Output (SVC 55)
   - Unit Allocation (IEFAB435)
   - IBM ACS Processing Routine

Environments

The environment indicates what allocation action is causing BrightStor CA-Allocate code to be entered. Depending on the environment, the ASR should take different actions. For example, the NEWNAME variable is checked only for a RENAME operation, since only RENAME causes a new name to be created. Likewise, for a new space allocation you check different variables and take different actions than for a scratch operation.

Twelve ASR environments are used during MVS data set operations:

1. **ACS Environment**
2. ALLOC Environment
3. DEFINE Environment
4. EOV Environment
5. EOV_VSAM Environment
6. EXTEND Environment
7. OLD Environment
8. **VDSPROG ASR and the QUOTA** Environment
9. RELEASE Environment
10. RELEASE Environment
11. SCRATCH Environment
12. SPACE Environment

Two ASR environments are used while the Quota Table is rebuilt during a QREBUILD operation. These environments receive control from the Started Task:

1. "**QREBUILD ASR and Environment**"
2. "**QSCAN ASR and Environment**"

The code is entered at the following points in MVS data set processing:

TABLE 2-2 Where the Code Gets Control

WHERE	ENVIRONMENTS	Data Set TYPES
In front of the Common Allocation routine (SVC 99)	ALLOC, OLD	Non-VSAM and VSAM (JCL define only)
IGDACS00	ACS	All
IGGPRE00 (IBM-provided exit point)	RELEASE, RENAME, SCRATCH, SPACE	Non-VSAM only
	EXTEND, QUOTA	Non-VSAM and VSAM
IGGPOST0 (IBM-provided exit point)	none	Non-VSAM and VSAM
In front of Catalog Management (SVC 26)	DEFINE, EOV_VSAM	VSAM only
End-of-Volume DASD Output (SVC 55)	EOV	Non-VSAM only.

The above information is summarized in a slightly different manner, see the variable description "**VAMENVIR**" in the chapter "**Implementation**."

Below is a brief description of each of the environments.

ACS Environment

This environment receives control for any Automatic Class Selection (ACS) processing that is requested by any system component. ACS processing is used to select or remove any of the four SMS constructs shown below. For more information about the four IBM SMS constructs, refer to the IBM manual *DFSMSdfp Storage Administration Reference.*

- Data class — DATACLAS
- Storage class — STORCLAS
- Management class — MGMTCLAS
- Storage group — STORGRP

BrightStor CA-Allocate receives control after the normal IBM ACS processing. It then runs through the user's ASR routine. The ASR routine can look at, set, or remove any of the four SMS constructs. The SMS constructs may have been set by either the original user request, or the IBM ACS processing (if any). No IBM ACS processing is required. The ACS environment can act as a direct replacement for IBM's ACS routines. This replacement lets the Storage Administrator use the greater power of the ASR rules language, and provides a single point for control of SMS and non-SMS-managed allocations.

Note: IBM procedures may require some minimum ACS routine setup. Refer to your IBM documentation.

All four constructs in the ASR rules can be set for each request that is made for ACS processing. If a particular construct is not requested, setting it has no effect. It is usually best to have one set of ACS logic in your ASR rules that sets all the constructs needed, all at one time. Doing so simplifies rules writing.

Return and reason code information can also be set by ASR rules, to be returned to the ACS requestor. The variables &ACSRC and &ACSRSN contain the values of the return code and reason code respectively. Upon entry to the ACS environment these two variables contain the values set by the IBM ACS processing. They can be looked at and overridden.

For example, if the JCL DATACLAS is bad, and no IBM ACS processing is involved, at entry into the ACS environment, &ACSRC sets to return code 8 and &ACSRSN sets to reason code 1014. These return and reason codes are documented in the IBM manual *DFSMSdfp Diagnosis Reference.* Refer to the sections “SMS Subsystem Interface Return Codes” and “ACS Services Reason Codes.”

To keep the allocation from failing, the ASR routine can set a valid data class name, or remove the bad data class name by setting it to null, set variables &ACSRC and &ACSRSN to zero, and exit the ASR with an EXIT CODE(0).

Any system component that requests ACS processing for any of the four SMS constructs sees the returned construct names just as if they had been set by the normal IBM ACS processing.

In a single request for ACS processing, a system component can request that any or all of the four SMS constructs be returned. The following list shows the most common requestors of ACS processing, and shows the constructs that each requests.

BATCH JCL ACS activity - This type of activity can be for VSAM or non-VSAM., including dynamic allocations (SVC 99), such as TSO ALLOC, ISPF 3.2, and others. One ACS call is made. This call requests all four constructs.

IDCAMS ACS activity - This type of activity can be for VSAM or non-VSAM, including File-Aid VSAM allocations. For SMS or non-SMS-managed the system makes 1 ACS call for the 3 classes to determine if a data set is SMS-managed. For SMS-managed only, the system makes 1 additional ACS call for the Storage Group only.

DFDSS ACS activity - This type of activity can be for VSAM or non-VSAM. For SMS or non-SMS-managed, the system makes 1 ACS call for STORCLAS and MGMTCLAS, but no DATACLAS is requested. For SMS-managed only, the system makes 1 additional ACS call for Storage Group only. The ACSENVIR variable is equal to “RECOVER.”

BrightStor CA-Disk Backup and Restore ACS activity - For non-VSAM it works the same as batch JCL and dynamic allocation activity described above. For VSAM it works the same as IDCAMS activity described above.

DFHSM ACS activity - This activity can be for VSAM or non-VSAM and includes RECOVER and RECALL. For SMS or non-SMS-managed, the system makes one ACS call for STORCLAS and MGMTCLAS only, but no DATACLAS is requested. For all but PO RECALLs it works as the same as DFDSS above. During a PO RECALL, ACS is entered one time for the &DSN real name and a second time for &DSN HSM name.

FIGURE 2-1 Sample ACS ASR

```
Menu     Utilities    Compilers    Help
------------------------------------------------------------------------------
BROWSE             .ASR.PARMLIB(ACS)  - 01.02          Line 00000000 Col 001 080
Command ==> _____________________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************
IF &VAMENVIR = 'ACS' THEN
    DO
       SET &DATACLAS = ''               /* DEFAULT TO NO DC           */
       IF  &DSORG = 'PO' THEN
          SET &DATACLAS = 'PODC'        /* DEFAULT FOR PO             */
       IF %HLQ = 'SYS1' THEN
          SET &STORCLAS = ' '           /* NOT SMS-MANAGED            */
       ELSE DO
            SET &SC = 'SMSVOL1'         /* SMS-MANAGED                */
            SET &MC = 'MANCL1'          /* MANAGEMENT CLASS           */
            SET &SG = 'SMSGRP1'         /* SMS STORAGE GROUP          */
          END
       SET &ACSRC = 0                   /* GOOD RETURN CODE           */
       SET &ACSRSN = 0                  /* GOOD RETURN CODE           */
       EXIT CODE(&ACSRC)                /* GOOD RETURN CODE           */
    END
******************************** Bottom of  Data ********************************
```

ALLOC Environment

This environment is entered for every new non-VSAM space allocation request, either batch or dynamic (SVC 99), and VSAM data sets allocated via SVC 99 (for example, for JCL VSAM defines, use &DSORG to detect VSAM). This environment is NOT entered for direct DADSM space requests (SVC 32) used by some utilities like IBM's IEHMOVE. The SPACE environment, described below, is entered for direct SVC 32 requests.

The ALLOC environment is special and provides the most powerful functions because it is not restricted to the capabilities of the IBM-provided DADSM pre-allocation exit (IGGPRE00), as are all of the rest of the environments.

Most of the output variables (changeable variables) may be changed in the ALLOC environment. See **TABLE 4-3 List of Variables**, for a list of output variables that can change. Figure 2-2 shows an example of ALLOC Environment ASR code. See also the section Enhanced DATACLAS Support later in this chapter.

FIGURE 2-2 Sample ALLOC Environment ASR

```
Menu     Utilities    Compilers    Help
------------------------------------------------------------------------
BROWSE          .ASR.PARMLIB(ALLOC)  - 01.03          Line 00000000 Col 001 080
Command ==> ____________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************
IF &VAMENVIR = 'ALLOC' THEN
    DO
       IF &DSTYPE = 'PERM' AND            /*     FOR PERMANENT DATA SETS    */
          &DSORG = 'PS' THEN              /*       THAT ARE SEQUENTIAL      */
             SET &RLSE = 'Y'              /* FREE UNUSED SPACE AT CLOSE     */
       IF &DSTYPE = 'TEMP' THEN           /* TEMPORARY DATA SETS ARE TO BE  */
          SET &STORGRP = 'SCRATCH'        /* REDIRECTED TO SCRATCH POOL     */
       ELSE
          IF &PGM = 'ADSMI002' AND        /* IF CA-DISK IS EXECUTING AND    */
             &DSN = *.DMS*.D*.T* THEN     /* THE DATA SET IS AN ARCHIVE     */
               SET &STORGRP = 'ARCHIVE'   /* REDIRECT TO THE ARCHIVE POOL   */
          ELSE
               SET &STORGRP =  'PRIMARY',  /* ALL OTHERS GO TO PRIMARY POOL  */
                               'OVERFLOW'  /* ANYEXTRA TO OVERFLOW POOL      */
    END
******************************** Bottom of  Data ********************************
```

One of the most important output variables (from a pooling standpoint) is the storage group (STORGRP, SG) variable. For a non-SMS-managed allocation, setting a storage group causes BrightStor CA-Allocate to go through its automatic volume selection routines, which select the best volumes from the pools of volumes that have been defined in the VDSTORGP member.

See also the section Enhanced DATACLAS Support later in this chapter.

All allocations that are “seen” by the ALLOC environment are also “seen” by the SPACE environment, unless they are explicitly failed in the allocation in the ASR with an EXIT CODE(4) statement.

DEFINE Environment

This environment is entered for every VSAM DEFINE request made to Catalog Management (SVC 26) for a Cluster or Alternate Index.

Note: When this environment is entered for JCL defined VSAM data sets (SVC 99 followed by SVC 26), any redirection attempt is ignored. To redirect this allocation, use the following ASR Statement in the ALLOC environment: &MODULE EQ ‘IEFIIC’. Doing so detects JCL defined VSAM. PTF LO98735 strengthened this requirement to include any SVC 99 request to define a VSAM data set. These allocations must be re-directed in the ALLOC environment. An example of this type of allocation is the IBM SYSTEM LOGGER logstream data sets.

Most output variables can be changed within the DEFINE environment. See **TABLE 4-3 List of Variables** in the chapter “**Implementation**” for a complete list of output variables that can change within this environment. See also the section Enhanced DATACLAS Support later in this chapter.

The most useful output variables (from a pooling standpoint) are the STORGRP, STORGRPD, and STORGRPI variables. These are the Storage Group variables, and they instruct BrightStor CA-Allocate to do volume selection from the volumes found within these pools (and within the UNIT specified, if any). The STORGRPD and STORGRPI variables are provided to allow a separate storage group to be set for the DATA and INDEX components of a KSDS. Figure 2-3 shows an example of DEFINE Environment ASR code.

FIGURE 2-3 Sample DEFINE Environment ASR

```
Menu     Utilities    Compilers    Help
------------------------------------------------------------------------
BROWSE          .ASR.PARMLIB(DEFINE)  - 01.03          Line 00000000 Col 001 080
Command ==> _____________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************
IF &VAMENVIR = 'DEFINE' THEN
   DO
     SET &CATONDEFOK = 'N'              /* DISALLOW CAT NAME ON DEFINE    */
     SET &STEPCATOK = 'N'               /* DISALLOW STEPCAT (IF ANY)      */
     SET &JOBCATOK = 'N'                /* DISALLOW JOBCAT (IF ANY)       */
     SET &OWNERID = &USER               /* OWNER OF VSAM DATA SETS        */
     IF &HLQ = 'ONLINE'THEN             /* IF ONLINE DATA SET             */
          DO
            SET &INDEXSEP = 'Y'         /* PUT INDEX COMPONENT ON A       */
            SET &STORGRP1 = 'CACHEDB'   /* HIGH PERFORMANCE VOLUME AND    */
            SET &STORGRPD = 'DATABASE'  /* REDIRECT DATA COMPONENT        */
          END
     ELSE                               /* ALL OTHER VSAM DATA SETS       */
         IF &HLQ ¬= SYS%                /* EXCEPT FOR SYSTEM ONES         */
         THEN SET &SG =  'PRIMARY',     /* ARE TO BE REDIRECTED           */
                         'OVERFLOW'
   END
******************************** Bottom of  Data ********************************
```

EOV Environment

Within the EOV environment, volumes can be dynamically added to a non-VSAM output DASD data set that has exhausted its space allocation. For more information about VSAM data sets, see the section EOV_VSAM Environment in this chapter.

This environment is entered whenever a job is about to fail for a system B37 or E37 type abend condition. As with the EXTEND environment, EOV is entered only for data sets that have a secondary space amount defined. BrightStor CA-Allocate becomes involved because the operating system tried to allocate secondary space to the data set but failed. If no secondary space amount was coded in the initial allocation JCL, the operating system would never try to add space. Some situations that can cause the EOV environment to be entered include:

- The current output volume is full and there are no more volumes allocated to the data set.
- The current output volume does not have enough space to allocate the requested amount of secondary space and there are no more volumes allocated to the data set.
- The data set already has 16 extents allocated on the current volume and there are no more volumes allocated to the data set.

The EOV environment provides the ability to control and save the job by coding ASR statements that set the STORGRP or POOLSUB variable. Any defined STORGRP name that is appropriate for the current situation may be used. Non-SMS-managed data sets are not restricted at end-of-volume time to the STORGRP specified in the original allocation. When a STORGRP is set for non-SMS-managed data sets, an attempt is made to pick a volume from the storage pool that:

- Is not currently allocated to the data set.
- Is of the same device type.
- Has sufficient free space to hold the requested allocation amount.
- Maintains free-space threshold requirements.

BrightStor CA-Allocate tries to maintain the percentage of free space defined for a volume in the VDSTORGP parameter file. This percentage is referred to as the free-space threshold and is specified in the VDSTORGP parameter file. Volumes that can be used for allocating the required space amount are seen as being either first or second choice volumes.

- First choice volume — The required space amount can be allocated without exceeding the free-space requirement.
- Second choice volume — The free-space requirement must be exceeded to allocate the required space amount.

The EOV_VSAM environment provides the ability to control and save the job by coding ASR statements that set the STORGRP variable. Any defined STORGRP name that is appropriate for the current situation may be used. Non-SMS-managed data sets are not restricted at end-of-volume time to the STORGRP specified in the original allocation. When a STORGRP is set for non-SMS-managed data sets, an attemp is made to pick a volume from the storage pool that:

- Is not currently allocated to the data set.
- Is of the same device type.
- Has sufficient free space to hold the requested allocationamount.
- Maintains free-space threshold requirements.

If no first choice volume can be found within the storage group, a second choice volume is selected. If BrightStor CA-Allocate cannot find a volume within the storage group that can satisfy the requested space amount, it cannot add a volume. In this case the allocation proceeds to the original B37 or E37 condition.

Since volumes can be dynamically added on demand, the Storage Administrator can select to force initial multi-volume allocations to a single volume (SET &NVOL=1, &NUNIT=1 in the ALLOC environment). If a data set runs out of space, BrightStor CA-Allocate can add volumes as needed.

For SMS-managed data sets, the volume is picked by SMS from the SMS STORGRP associated with the data set. For more information, see the section Using EOV and EOV_VSAM Environments with a SMS-Managed Data Set in this chapter.

SnapShot

SnapShot target data sets should be excluded from BrightStor CA-Allocate EOV environment. According to IBM, the SIBBATCH program cannot handle any space change or volume redirection done by any other OEM product at this time.

```
IF &VAMENVIR = 'EOV' AND &PGM = 'SIBBATCH' THEN EXIT CODE(0).
```

EOV_VSAM Environment

Within the EOV_VSAM environment, candidate volumes can be dynamically added to a VSAM output data set. For non-VSAM data sets, see the section EOV Environment in this chapter. The EOV_VSAM environment is entered whenever the VSAM EOV requests the next candidate volume and VSAM indicates that there are no more candidates. It becomes involved because the operating system tried to allocate a new output volume and could not. Some situations that can cause the EOV_VSAM environment to be entered include:

- The current output volume is full and no more candidate volumes are available.
- The current output volume does not have enough space to allocate the requested amount of secondary space and no more candidate volumes are available.

BrightStor CA-Allocate tries to maintain the percentage of free space that has been defined for a volume. This percentage is referred to as the free-space threshold and is specified in the VDSTORGP parameter file. Volumes that could be used for allocating the required space amount are seen as being either first or second choice volumes.

- First choice volume — The required space amount can be allocated without exceeding the free-space requirement.
- Second choice volume — The freespace requirement must be exceeded to allocate the required space amount.

If no first choice volume can be found within the storage group, a second choice volume is selected. If it cannot find a volume within the storage group that can satisfy the requested space amount, it cannot add a volume.

Since volumes can be dynamically added on demand, the Storage Administrator may choose to force initial multi-volume VSAM defines to a single volume (SET &NVOLD=1 in the DEFINE environment). If a data set runs out of space, BrightStor CA-Allocate can then add volumes as needed.

For SMS-managed data sets, the volume is picked by SMS from the SMS STORGRP associated with the data set. For more information, see the section Using EOV and EOV_VSAM Environments with a SMS-Managed Data Set in this chapter.

SnapShot

SnapShot target data sets should be excluded from BrightStor CA-Allocate EOV_VSAM environment. According to IBM, the SIBBATCH program cannot handle any space change or volume redirection done by any other OEM product at this time.

For EOV_VSAM environment screening, add:

```
IF &VAMENVIR = 'EOV_VSAM' AND &PGM = 'SIBBATCH' THEN EXIT CODE(0)
```

EXTEND Environment

Non-VSAM Processing in EXTEND

This environment is entered whenever a data set needs to grow out to another extent (that is, needs more disk space). NonVSAM data sets need a secondary amount in order to continue to grow after they have exhausted their primary amount. The EXTEND Environment is launched prior to the allocation of each additional extent. To get to this environment, the data set must either have been originally allocated with secondary amount or BrightStor CA-Allocate must be running in a **PLSZSEC** (Y) configuration. See the EOV Support for Zero Secondary section later in this Chapter and the **SECONDARY_HAD_ZERO** variable description in the chapter "Implementation" for how to avoid D37 abends and how to then detect that they had been avoided.

In this environment the Storage Administrator can choose to check the current number of extents to which the data set has grown, and perhaps increase the secondary allocation amount to decrease the chance of getting an abend for an allocation that is too small. Figure 2-4 shows this. Another way to avoid an out-of-space abend is to dynamically add a volume to the data set in the EOV environment.

FIGURE 2-4 Sample EXTEND Environment ASR for Non-VSAM

```
Menu     Utilities    Compilers    Help
------------------------------------------------------------------------
BROWSE          .ASR.PARMLIB(VDSPROG)  - 01.04          Line 00000000 Col 001 080
Command ==> _____________________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************

IF &VAMENVIR = 'EXTEND'      /*          NON-VSAM EXTEND                    */
   AND &EXTENTS > 8 THEN                 /* IF RUNNING OUT OF EXTENTS,      */
     SET &SECONDARY = 2 * &SECONDARY     /* INCREASE SIZE OF ALLOCATION     */
      SET &LSPACE_VOLUME = &LSPACE_VOLUME  /*    MAKE SURE THERE            */
          IF &LSPACE_RETURN_CODE = 0 THEN  /*    IS ENOUGH ROOM             */
         DO              IF &SPACTYPE = 'CYL' THEN
            DO
              IF &SECONDARY GT &LEC
                 SET &SECONDARY EQ &LEC
            END
         END
******************************** Bottom of  Data ******************************
```

Note: If an extend request is rejected in the ASR, the job requesting the extension is abended with a system code E37-0C. This is a restriction of the DADSM IGGPRE00 exit.

Extend Environment SnapShot

SnapShot target data sets should be excluded from BrightStor CA-Allocate VSAM Extend environment. According to IBM, the SIBBATCH program cannot handle any space change or volume redirection done by any other OEM product at this time.

For VSAM EXTEND environment, apply the GA PTF for PI 369129 (SS04708).

VSAM Processing in EXTEND

This environment is entered whenever a cluster needs to grow into another extent. However, this support is deactivated by default. For activation procedures, see **PLSOPT11** in the section **VKGPARMS Parameters** of the chapter "**Implementation**." MVS End of Volume (EOV) processing is normally the cause for entering this environment, as long as the cluster definition has secondary allocation.

Note: The difference between this support and that for non-VSAM, described in the section Non-VSAM Processing in EXTEND in this chapter, is that with non-VSAM, the space allocated for a *new volume* extent is associated with the data set's secondary allocation amount.

Note: VSAM, typically the cluster's **primary** allocation amount is used for *new volume* extents. This can be changed, for SMS-managed data sets only, using the ADDITIONAL VOLUME AMT DATA CLASS attribute.

Restrictions to VSAM Processing in EXTEND - The following VSAM attributes are not supported:

- VSAM clusters defined to IMS
- KSDS clusters defined with either the KEYRANGE or IMBED attributes
- VSAM clusters associated with SNAPSHOT copies
- VSAM allocations made by IBM's Media Manger Services, including, but not necessarily limited to, those of the DB2 DBM1 STC and BMC's LOAD+ Utility.

Variables - In order to support VSAM, two new ASR variables are now available. The following is a list of these variables:

TABLE 2-3 VSAM EXTEND Variables

Variable Name	Alias
&**EXTENDCOMP**	EC
&**EXTENDVOL**	EV

These variables allow you to dynamically alter primary and/or secondary component allocation values. They also give you the choice of extending onto the current volume or, after successfully recovering from an end-of-volume condition, onto a new volume. Figure 2-5 (described in 3 sections) shows this. Another way to avoid an out-of-space abend is to dynamically add a volume to the data set in the EOV environment.

FIGURE 2-5 Sample EXTEND Environment ASR for VSAM — 1 of 3.

```
Menu     Utilities    Compilers    Help
------------------------------------------------------------------------
BROWSE          .ASR.PARMLIB(NEWEXTVS)  - 01.06          Line 00000000 Col 001
080
Command ==> _____________________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************
   IF &VAMENVIR = 'EXTEND' AND &DSORG = 'VS' AND &EXTENTS > 50 THEN DO
     SET &LSPACE_VOLUME = &VOLSER
     IF &LSPACE_RETURN_CODE = 0 THEN DO
          COPYBOOK 'CURRVOL'
          COPYBOOK 'NEWVOL'
                                                                    END
                                    END
******************************** Bottom of  Data ********************************
```

```
Menu     Utilities    Compilers    Help
------------------------------------------------------------------------
BROWSE          .ASR.PARMLIB(CURRVOL)  - 01.02          Line 00000000 Col 001 080
Command ==> _____________________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************
   IF &EXTENDVOL = 'CURRENT' THEN
   DO
     IF &EXTENDCOMP = 'DATA' THEN
     DO
       IF &SPACTYPED = 'CYL' THEN SET &SECONDARYD = &LARGEST_EXTCYL
       IF &SPACTYPED = 'TRK' THEN SET &SECONDARYD = &LARGEST_EXTTRK
     END
     IF &EXTENDCOMP = 'INDEX' THEN
     DO
       IF &SPACTYPEI = 'CYL' THEN SET &SECONDARYI = &LARGEST_EXTCYL
       IF &SPACTYPEI = 'TRK' THEN SET &SECONDARYI = &LARGEST_EXTTRK
     END
   END
******************************** Bottom of  Data ********************************
```

```
Menu     Utilities    Compilers    Help
------------------------------------------------------------------------
BROWSE          .ASR.PARMLIB(NEWVOL)  - 01.02          Line 00000000 Col 001 080
Command ==> _____________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************
   IF &EXTENDVOL = 'NEW' THEN
   DO
     IF &EXTENDCOMP = 'DATA' THEN
     DO
       IF &SPACTYPED = 'CYL' THEN SET &PRIMARYD = &LARGEST_EXTCYL
       IF &SPACTYPED = 'TRK' THEN SET &PRIMARYD = &LARGEST_EXTTRK
     END
     IF &EXTENDCOMP = 'INDEX' THEN
     DO
       IF &SPACTYPEI = 'CYL' THEN SET &PRIMARYI = &LARGEST_EXTCYL
       IF &SPACTYPEI = 'TRK' THEN SET &PRIMARYI = &LARGEST_EXTTRK
     END
   END
******************************** Bottom of  Data ********************************
```

OLD Environment

This environment is entered for any batch job's allocation request for an old, cataloged data set that may be specifying invalid unit information which, if ignored, causes MVS to terminate the request with an **"IEF210I UNIT FIELD SPECIFIES INCORRECT DEVICE NAME"** JCL error. This condition most often occurs when BrightStor CA-Allocate is instructed to redirect the original allocation to a different device type without changing the UNIT type in the JCL to reflect the change.

For performance reasons, this support is limited to only cataloged data sets, and in addition the following data set types are automatically bypassed:

- VSAM
- SMS-managed
- Non-SMS-managed that do not have any UNIT value
- Archived or migrated data sets

Two variables are available.

1. UNITMISMATCH (UMM) is a "yes" or "no" flag indicating whether or not there is an inconsistency between what is specified in the JCL and what is in the catalog entry.
2. FIXUNITMISMATCH (FUMM) contains the name of a valid UNIT that prevents the JCL error. For disk data sets, FUMM always contains 'SYSALLDA'. For tape data sets, FUMM contains the generic unit name associated with the device type in the catalog entry (for example, '3480', '3420', and so on).

Note: The OLD environment compares the unit information in the catalog against what is coded in the JCL. If BrightStor CA-Allocate thinks there is a difference, it is a UNITMISMATCH (UMM). When you originally allocate a data set with an esoteric (for example, VAULT) UNIT=VAULT, the generic, rather than the esoteric, is stored in the catalog (3490). If you reference it again with the esoteric, then the UNIT=VAULT (esoteric) does not equal the CATALOGUED UNIT=3490 (generic), thus the UNITMISMATCH condition and it is resolved to a generic name (3490).

Figure 2-6 shows an example of an Allocation Selection Routine for the OLD Environment that detects and corrects a unit mismatch condition.

FIGURE 2-6 Sample OLD Environment ASR

```
Menu     Utilities    Compilers    Help
------------------------------------------------------------------------
BROWSE          .ASR.PARMLIB(VDSPROG)  - 01.00          Line 00000000 Col 001 080
Command ==> _____________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************

IF &VAMENVIR = 'EXTEND'
   AND &EXTENTS > 8 THEN              /* IF RUNNING OUT OF EXTENTS,        */
     SET &SECONDARY = 2 * &SECONDARY  /* INCREASE SIZE OF ALLOCATION       */

IF &VAMENVIR = 'OLD'
THEN
DO
IF &UNITMISMATCH = 'Y'           /* IF UNIT MISMATCH DETECTED */
THEN                             /* CORRECT IT TO PREVENT */
SET &UNIT = &FIXUNITMISMATCH     /* JCL ERROR & JOB FAILURE */
IF &NUNIT > &CAT_VOL_COUNT       /* EXCESS UNIT-COUNT IN JCL*/
THEN                             /* REDUCE TO INSURE */
SET &NUNIT = &CAT_VOL_COUNT      /* EOV SUPPORT ELIGIBILITY*/
END
IF &VAMENVIR = 'RELEASE' THEN
   DO
     IF &HLQ = SYS% THEN                 /* IF SYSTEM DATA SET             */
          DO
            WRITE 'YOU MAY NOT RELEASE SPACE FROM A SYSTEM DATA SET'
            EXIT CODE(8)                 /*  THEN DISALLOW THE RELEASE     */
          END
   END
****************************** Bottom of  Data ******************************
```

In addition to the UMM and FUMM variables, the OLD Environment has access to most of the input-only ones available in the ALLOC Environment, as shown in **TABLE 4-3 List of Variables** in the chapter "**Implementation**." Like the UNIT parameter shown above in Figure 2-6, the Allocation Selection Routine can examine others (for example, Data set Name, User, Job Name) in order to decide whether to fix the unit mismatch.

If volumes are specified in the allocation request, they must match the ones in the catalog entry exactly, in both number and sequence. When they do not, the allocation request is considered to be for an uncataloged data set, and the OLD Environment is bypassed. If it is not possible to determine the catalog status of the data set, as happens with specific allocation requests on DFP Release 2 systems, the OLD Environment is bypassed.

Uncataloged data sets are skipped in order to avoid the significant overhead involved in accessing individual DASD VTOCs or Tape Management System control files.

Note: Results will be UNPREDICTABLE if &UNIT is set to anything but &FIXUNITMISMATCH in the OLD Environment.

Another feature of the OLD Environment is to allow reduction of unit-count values specified in the JCL UNIT parameter that might otherwise render allocations ineligible for NonVSAM EOV Recovery. Details can be found in the **CAT_VOL_COUNT** variable description. Figure 2-6 includes an example of applicable ASR code.

QREBUILD, QSCAN, QUOTA Environments

These environments, applicable only to the Quota selectable unit, are described in the section **Allocation Selection Routines and Environments** of the chapter "**Quota**."

RELEASE Environment

This environment is entered whenever a request is made to release unused space from a non-VSAM data set. The request can be made from a batch job with the RLSE parameter of the DD statement, or in the TSO ALLOC command using the RELEASE operand.

In this environment, the Storage Administrator can choose to stop the release of space from certain data sets through the ASR shown below in Figure 2-7.

FIGURE 2-7 Sample RELEASE Environment ASR

```
Menu     Utilities    Compilers    Help
------------------------------------------------------------------------
BROWSE          .ASR.PARMLIB(VDSPROG)  - 01.00          Line 00000000 Col 001 080
Command ==> _____________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************

IF &VAMENVIR = 'EXTEND'
   AND &EXTENTS > 8 THEN              /* IF RUNNING OUT OF EXTENTS,        */
     SET &SECONDARY = 2 * &SECONDARY  /* INCREASE SIZE OF ALLOCATION       */
IF &VAMENVIR = 'OLD'
   AND &UNITMISMATCH = 'Y'              /* IF UNIT MISMATCH DETECTED       */
         THEN                           /*    CORRECT IT TO PREVENT        */
         SET &UNIT = &FIXUNITMISMATCH/*  JCL ERROR & JOB FAILURE           */
IF &VAMENVIR = 'RELEASE' THEN
   DO
     IF &HLQ = SYS% THEN                /* IF SYSTEM DATA SET              */
         DO
           WRITE 'YOU MAY NOT RELEASE SPACE FROM A SYSTEM DATA SET'
           EXIT CODE(8)                 /* THEN DISALLOW THE RELEASE       */
         END
   END
****************************** Bottom of  Data ******************************
```

Note: If the decision is made to stop the release, it should issue an appropriate WRITE statement to tell the offender why the release was denied. The IBM system messages are not very clear and do not point to BrightStor CA-Allocate.

RENAME Environment

This environment is entered whenever a request is made to rename a non-VSAM data set. If an installation wants to maintain control over which data set names can be allocated to which volumes, this environment can be used to prevent a user from renaming a good data set name to a bad data set name, as shown below in Figure 2-8.

FIGURE 2-8. Sample RENAME Environment ASR

```
Menu     Utilities    Compilers    Help
------------------------------------------------------------------------
BROWSE          .ASR.PARMLIB(VDSPROG)  - 01.01          Line 00000000 Col 001 080
Command ==> _____________________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************
IF &VAMENVIR = 'RENAME'
   DO
     IF &HLQ = SYS% THEN                    /* IF SYSTEM DATA SET          */
          DO
            WRITE 'YOU MAY NOT RENAME A SYSTEM DATA SET'
            EXIT CODE(8)                  /*  THEN DISALLOW THE RENAME       */
          END
   END
IF &VAMENVIR = 'SCRATCH' THEN
   DO
     IF &HLQ = SYS% THEN                  /* IF SYSTEM DATA SET             */
          DO
            WRITE 'YOU MAY NOT SCRATCH A SYSTEM DATA SET'
            EXIT CODE(8)                  /*  THEN DISALLOW THE SCRATCH      */
          END
   END
******************************** Bottom of  Data ********************************
```

Note: If the decision is made to stop the rename, it should issue an appropriate WRITE statement to tell the offender why the rename was denied. The IBM system messages are not very clear and do not point to BrightStor CA-Allocate.

SCRATCH Environment

This environment is entered whenever a request is made to delete a non-VSAM data set. In this environment, the Storage Administrator can choose to stop the deletion of certain important data sets through the ASR, as shown in Figure 2-9.

For more information about the SCRATCH environment, see the section **SMS-Managed GDGs** of the chapter "**Introduction**."

FIGURE 2-9 Sample SCRATCH Environment ASR

```
Menu     Utilities    Compilers    Help
------------------------------------------------------------------------
BROWSE          .ASR.PARMLIB(VDSPROG)  - 01.01          Line 00000000 Col 001 080
Command ==> _____________________________________________________ Scroll ==> CSR

********************************* Top of  Data ********************************
IF &VAMENVIR = 'RENAME'
   DO
     IF &HLQ = SYS% THEN                    /* IF SYSTEM DATA SET          */
          DO
            WRITE 'YOU MAY NOT RENAME A SYSTEM DATA SET'
            EXIT CODE(8)                   /*  THEN DISALLOW THE RENAME       */
          END
   END
IF &VAMENVIR = 'SCRATCH' THEN
   DO
     IF &HLQ = SYS% THEN                   /* IF SYSTEM DATA SET             */
          DO
            WRITE 'YOU MAY NOT SCRATCH A SYSTEM DATA SET'
            EXIT CODE(8)                   /*  THEN DISALLOW THE SCRATCH      */
          END
   END
******************************** Bottom of  Data ******************************
```

Note: If the decision is made to stop the scratch, it should issue an appropriate WRITE statement to tell the offender why the scratch was denied. The IBM system messages are not very clear and do not point to BrightStor CA-Allocate.

SPACE Environment

This environment is entered for every new non-VSAM space allocation request, either batch, dynamic (SVC 99), or direct DADSM space request (SVC 32). This environment is required in addition to the ALLOC environment, because some utilities (for example, IBM's IEHMOVE) directly issue an SVC 32 (DADSM space request) to allocate disk space, and bypass the normal flow of allocations. Allocations made by these utilities are not “seen” by the ALLOC environment, but can be checked in the SPACE environment. Only a limited number of output variables can be set in the SPACE environment, as shown in **TABLE 4-3 List of Variables** in the chapter “**Implementation**.” Most notably, it is not possible to select volumes in the SPACE environment. That is, the ASR cannot set a storage group (STORGRP).

The only function that IBM's IGGPRE00 exit allows at this point is to accept the volume (EXIT CODE(0)), reject the current volume (EXIT CODE(4)), or fail the allocation request immediately (EXIT CODE(8)). Figure 2-10 shows this situation. Any specific volume allocation fails if any volume was rejected (so EXIT CODE(4) and EXIT CODE(8) acts the same). For a non-specific volume allocation (no VOL=SER= or equivalent), MVS presents a list of all volumes within the specified unit name (UNIT=) that are mounted with a mount attribute of STORAGE.

FIGURE 2-10. Sample SPACE Environment ASR

```
Menu     Utilities    Compilers    Help
------------------------------------------------------------------------
BROWSE          .ASR.PARMLIB(SPACE)  - 01.01          Line 00000000 Col 001 080
Command ==> _____________________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************
IF &VAMENVIR = 'SPACE' THEN
   DO                                      /* SPACE ENVIRONMENT          */
     IF &HLQ = SYS% THEN                   /* IF SYSTEM DATA SET         */
          DO
            WRITE 'YOU MAY NOT RESTORE OR COPY A SYSTEM DATA SET'
            EXIT CODE(8)                   /* CANCEL THE ALLOCATTION     */
          END
     IF &ANYVOL = SY* THEN                 /* IF SYSTEM VOLUME           */
          DO
            WRITE 'YOU MAY NOT RESTORE OR COPY A SYSTEM DATA SET'
            EXIT CODE(4)                   /* REJECT THE CURRENT VOLUME  */
          END
   END
****************************** Bottom of  Data ******************************
```

Note: If you set a storage group (STORGRP) in the ALLOC environment, BrightStor CA-Allocate makes the allocation request specific.

Note: MVS issues messages IEF344I and / or IGD17037I if the ASR exits with a return code of 8. To avoid confusing end users, document your action with an appropriate WRITE statement.

Note: Optional PI 601599 is required to have “EXIT CODE(8)” cause the allocation to be cancelled.

Note: When allocations are deferred via PLSOPT9 (Y), and the ASR exits with a non-zero return code, there will be no special MVS messages issued. It is therefore important, to avoid confusing end users, to have the ASR issue some sort of message indicating that it is choosing to deny this allocation.

Life Cycle of a Data Set

Table 2-4 shows the sequence of ASR Environments that are entered during the life cycle of a data set, from the time it is originally allocated, through open, close, extension, rename, and finally deletion, assuming the optional Quota feature is not installed. Table 7-1, “Life cycle of a data set with Quota,” shows the points in which the QUOTA Environment is entered during a similar data sets life cycle.

TABLE 2-4 Life cycle of a data set without Quota

OPERATION	non-VSAM	VSAM
Allocate a new single-volume data set.	ACS, ALLOC, and SPACE	ACS & DEFINE (cluster)
Write to the data set so that an extent is required.	EXTEND	EXTEND
Release unused space in the data set when it is closed.	RELEASE	not applicable
Rename the data set.	OLD and RENAME	not applicable
Write to the data set so that an additional volume is required.	EOV	EOV_VSAM (component)
Scratch the data set.	SCRATCH (volume 1) SCRATCH (volume 2)	not applicable

Volume Selection Processing Logic

BrightStor CA-Allocate involvement in selecting what volumes are selected for SMS-managed data sets is limited to specifying what DFSMS STORGRP IBM's SMS subsystem is to use. For non-SMS-managed allocations, it takes a more active role by choosing the exact volume that will be used. This volume selection processing logic is activated by setting the STORGRP variable to one or more storage group names. A list is then made of all volumes that are online for the unit name specified in the UNIT variable. If you specify a nonexistent unit or a specific unit address like 1A0, BrightStor CA-Allocate issues message VAM0030 and bypasses volume selection. Any volumes with no free dscb's or duplicate data set names will also be excluded from the volume selection process.

BrightStor CA-Allocate selects from the volumes that are in the first storage group. It then passes this list of volumes to the System Resource Manager (SRM). SRM selects the volume on the least-used channel with the fewest users. Next, it calls the security interface to see if the user is authorized to allocate a data set on the selected volume. If the security system indicates that the user is authorized, the selected volume is checked for enough space.

If the volume contains insufficient space for the primary quantity, it is deleted from the list and SRM is invoked again. If the volume contains sufficient space but the free-space threshold would be violated, the volume is marked as a second choice volume.

This process continues until the allocation has been satisfied or the first storage group is exhausted. The remainder of the storage groups in the STORGRP variable are checked, if necessary, one after another for first-choice volumes. If all storage groups are exhausted, BrightStor CA-Allocate uses second choice volumes in the order in which SRM selected them. PLSOPT22 can be used to change the volume selection process of the Second choice volumes. See the description of **PLSOPT22** in the chapter "**Implementation**."

End-of-volume Support

MVS JCL permits users to allocate multi-volume data sets. BrightStor CA-Allocate extends this support by allowing the ASR to change the allocation to or from a multi-volume request in the ALLOC and DEFINE environments. All data sets, whether multi-volume or not, can have special processing considerations:

- The data set may not actually use or need extra volumes coded in the initial allocation (wasted space to avoid a B37 abend).
- The user may not correctly estimate the amount of space or number of volumes needed to run a job. By the time the data set actually needs to go to a secondary extent or volume, that space may not be available. If space is unavailable, the job abends with a B37.
- Jobs that reference the data set may also allocate additional volumes (resources) even if the data set does not actually use them.
- Free-space thresholds are not maintained because the additional volumes are allocated before they are actually written to. By the time the data set spans to the additional volumes, the original amounts of free space are no longer applicable.

This EOV support enhances the process of controlling multi-volume as well as non-multi-volume data sets. From the BrightStor CA-Allocate point of view, it makes no difference whether the initial allocation was for multiple volumes or not. The EOV and EOV_VSAM environments get control when a data set has run out of space. Within these environments, more volumes can be dynamically added on an as-needed basis.

Using the EOV support means:

- There is no need to preallocate all required volumes. Additional volumes are picked as they are needed.
- B37 and E37 type abends can be prevented by adding an additional volume, allowing the job to continue.
- Free-space threshold requirements are easier to maintain, because these are considered when additional volumes are selected.

Technical Overview

EOV and EOV_VSAM environments get control when a data set has run out of space on the current volume and there are no more volumes from which to get space.

For those VSAM data sets that are created with only primary allocation and an EOV_VSAM environment is coded in the VDSPROG, BrightStor CA-Allocate allocates another volume using the primary allocation space. For certain VSAM data sets that do not have secondary space and you do not want them to enter the EOV_VSAM environment, exclude them from the BrightStor CA-Allocate allocation routine by checking for secondary space and then exit code (0). If excluded from BrightStor CA-Allocate and the data set exceeds the primary allocation, the IBM message IEF070I 203-204 is issued, revealing an error in EOV processing when attempts to extend were made but no secondary space was specified.

Setting the STORGRP within these environments allows the extension of an output data set onto multiple volumes, even when those volumes were not coded in the original allocation. Any defined STORGRP name that is appropriate for the current situation can be used. Non-SMS-managed data sets are not restricted at end-of-volume time to the STORGRP specified in the original allocation. When the ASR sets STORGRP, STORGRPD or STORGRPI for non-SMS-managed data sets, an attempts made to pick a volume from the storage group that:

- Is not currently allocated to the data set.
- Is of the same device type.
- Has sufficient free space to hold the requested allocation amount.
- Maintains free-space threshold requirements.

BrightStor CA-Allocate tries to maintain the percentage of free space that has been defined for a volume. This percentage is referred to as the free-space threshold. Available volumes are seen as being either first or second choice volumes.

- First choice volume — The required space amount can be allocated without exceeding the free-space requirement.
- Second choice volume — The free-space requirement must be exceeded to allocate the required space amount.

If no first choice volume can be found within the storage group, a second choice volume is selected.

Restrictions

Not all data sets can be extended to multi-volume data sets. For the following data set types and conditions, the EOV and EOV_VSAM environments are not entered:

- VIO data sets are allocated in page spaces, not volumes.
- IBM does not support multi-volume PO and PDSE data sets.
- Data sets that have either an ABEND or EOV exit coded in their DCB's exit parameter list are usually not processed. See the information on PIs **601765** and **601769** in the appendix "**Optional Maintenance**" for details on how these exclusions can be disabled.
- Data sets with more than one open DCB.

Understanding When EOV is Entered

The EOV environment allows output data sets to extend onto multiple volumes. Nothing seems more frustrating than watching a long running job abend because one of its output data sets has run out of space. The abend can be avoided within this environment, allowing the job to continue by allocating additional volumes.

The nature of multi-volume data sets is somewhat complex and as such requires the ASR writer to be careful about when such a situation should be corrected. With that in mind, some prescreening of the data sets is done to ensure that making them multi-volume does not cause additional problems. Consider the following JCL (albeit strange it is also perfectly valid):

```
//JOB       JOB
//STEP1     EXEC PGM=IEFBR14
//AFILE      DD DSN=THE.SAME.OLD.FILE,
//          DISP=(NEW,CATLG),VOL=SER=(PVOL01,PVOL02),UNIT=..,SPACE=..
//BFILE      DD DSN=THE.SAME.OLD.FILE,
//          DISP=(NEW,KEEP),VOL=SER=PVOL03,UNIT=..,SPACE=..
//CFILE      DD DSN=ANOTHER.FILE,
//          DISP=(NEW,KEEP),VOL=SER=PVOL03,UNIT=..,SPACE=..
//STEP2     EXEC PGM=IEBGENER
//SYSUT1     DD DSN=FILEIN,DISP=SHR
//SYSUT2     DD DSN=THE.SAME.OLD.FILE,DISP=OLD,VOL=SER=PVOL02,UNIT=..
//       ...
//STEP3     EXEC PGM=IEBGENER
//SYSUT1     DD DSN=FILEIN,DISP=SHR
//SYSUT2     DD DSN=THE.SAME.OLD.FILE,DISP=(OLD,CATLG),VOL=SER=PVOL03,UNIT=..
//       ...
//STEP4     EXEC PGM=IEBGENER
//SYSUT1     DD DSN=FILEIN,DISP=SHR
//SYSUT2     DD DSN=THE.SAME.OLD.FILE,
//          DISP=(NEW,CATLG),VOL=SER=PVOL04,UNIT=..,SPACE=..
//       ...
//STEP5     EXEC PGM=IEBGENER
//SYSUT1     DD DSN=FILEIN,DISP=SHR
//SYSUT2     DD DSN=THE.SAME.OLD.FILE,DISP=OLD
// ...
//STEP6     EXEC PGM=IEBGENER
//SYSUT1     DD DSN=FILEIN,DISP=SHR
//SYSUT2     DD DSN=ANOTHER.FILE,DISP=(OLD,CATLG),VOL=SER=PVOL03,UNIT=..
//       ...
//STEP7     EXEC PGM=IEBGENER
//SYSUT1     DD DSN=FILEIN,DISP=SHR
//SYSUT2     DD DSN=ABRAND.NEW.FILE,
//           DISP=(NEW,CATLG),VOL=SER=PVOL03,UNIT=..,SPACE=..
//       ...
//STEP8     EXEC PGM=IEBGENER
//SYSUT1     DD DSN=FILEIN,DISP=SHR
//SYSUT2     DD DSN=ANOTHER.BRAND.NEW.FILE,DISP=(NEW,KEEP),UNIT=..
//       ...
```

STEP1 - creates both cataloged and uncataloged versions of the same data set. Assuming that each of the output data sets runs out of space and needs additional volumes, on what basis is the EOV ASR be invoked? In the following paragraphs the filtering done by EOV is described for each step.

STEP2 - references the second volume of the cataloged data set created in STEP1. In this case, the EOV environment is not entered, since the data set is being referenced by UNIT and VOL, rather than through the catalog. If this data set were made multi-volume, the catalog would no longer point at the appropriate volumes. Future jobs or steps that referenced the data set would not be able to find the data set correctly.

STEP3 - is referencing the uncataloged data set created in STEP1. Additionally, when the step has completed, the data set is to be cataloged. Since the data set is already cataloged, the system returns a NOT CATLGD 2 at step termination. The EOV routines detects that a cataloged version of the data set exists and does not enter the EOV ASR, because no new volume can be cataloged, and so no future job can find it.

STEP4 - is attempting to create yet another cataloged version of the same file, created and cataloged by STEP1. Again the system returns a NOT CATLGD 2 at step termination since the data set is already cataloged. The EOV routines detect that the data set is already cataloged and does not enter the EOV ASR, because future jobs or steps are unable to find any added volume.

STEP5 - references the cataloged data set created in STEP1 and is not volume specific. In this case the system locates the data set via the catalog and plugs in the appropriate volumes automatically. This data set is processed by EOV since EOV can correctly re catalog the data set.

STEP6 - references the uncataloged data set allocated by STEP1, intending to catalog the data set at the end of the step. Since this data set is not currently cataloged, and it can be correctly cataloged at step termination, it is processed by EOV.

STEP7 - is creating a new data set to be cataloged at step termination. Since this data set is not currently cataloged, and it can be correctly cataloged at step termination, it is processed by EOV.

STEP8 - is creating a permanent uncataloged data set, which is eligible for EOV processing.

Using EOV and EOV_VSAM Environments with a SMS-Managed Data Set

When the EOV and EOV_VSAM environments are entered for a data set that is not SMS-managed, another volume can be added to the data set by having BrightStor CA-Allocate choose one (that is, by setting a STORGRP). When the EOV and EOV_VSAM environments are entered for a SMS-managed data set, BrightStor CA-Allocate only prepares a data set to receive another volume. SMS allocates another volume to the data set only if it can do so without violating SMS restrictions:

- SMS can only choose from the SMS STORGRP that is already associated with the data set. It cannot choose a volume that is outside of that STORGRP.
- If the data set is already allocated to every volume in that STORGRP, then it is not possible for another volume to be added, even though the EOV or EOV_VSAM environment specified that one should be added.

Note: For complete limits and restrictions on SMS-managed multi-volume data sets, refer to the appropriate IBM documentation.

SMS can allocate a candidate volume at initial allocation time for both VSAM and Non-VSAM data sets. The actual candidate volume, indicated by an “*” in the catalog entry's volume list, is not chosen until needed. When it is needed, SMS replaces the candidate volume with a real volume from within the SMS STORGRP that is associated with the data set. BrightStor CA-Allocate can take advantage of this process with its ability to add a candidate volume. It dynamically updates the necessary control blocks and the catalog to make it appear to SMS that the candidate volume was present when the original allocation occurred.

Any data set that has a STORCLAS not equal to null (&STORCLAS NE '') is SMS-managed. When processing SMS-managed data sets in the EOV and EOV_VSAM environments, the associated DATACLAS, MGMTCLAS, and STORCLAS construct information is available as read-only: it cannot be modified.

To add a candidate volume to an SMS-managed data set, either the &STORGRP variable must be set to the keyword “SMSEOV” (that is, SET &STORGRP = 'SMSEOV'), or the &POOLSUB variable set to 'Y', and an “EXIT CODE(0)” statement must be used. This causes BrightStor CA-Allocate to make a series of changes that make the data set appear to SMS as a one that has a candidate volume.

If the &STORGRP variable is set to any value other than 'SMSEOV', these changes are not made and the out of space condition occurs. This includes setting it to a valid storage group that has previously been defined to either SMS or BrightStor CA-Allocate. In addition, the EOV and EOV_VSAM environments cannot be exited with “EXIT CODE(x)”, where x is non-zero.

Figure 2-11 has sample ASR code for the EOV and EOV_VSAM environments.

FIGURE 2-11 Sample EOV ASR

```
Menu     Utilities    Compilers    Help
------------------------------------------------------------------------
BROWSE          .ASR.PARMLIB(EOV)  - 01.01          Line 00000000 Col 001 080
Command ==> _____________________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************
FILTLIST &SMSPROD INCLUDE('SCPROD01','SCMAST01')
FILTLIST &PROD INCLUDE(PROD.**,,ASTER.**)
IF &VAMENVIR = 'EOV' OR &VAMENVIR = 'EOV_VSAM' THEN
   DO
     IF &SC = &SMSPROD                       /*   SMS-MANAGED PRODUCTION    */
        OR &DSN = &PROD                      /*  NONSMS-MANAGED PRODUCTION  */
            THEN                             /*    GIVE IT ANOTHER VOLUME   */
                SET &POOLSUB = 'Y'           /*      FROM THE SAME POOL     */
     ELSE DO                                 /*   FOR NON PRODUCTION JOBS   */
             WRITE '&DSN IS NOT PRODUCTION AND OUT OF SPACE'
             EXIT CODE(4)                    /*     DO NOT ADD A VOLUME    */
        END
   END
******************************** Bottom of  Data ********************************
```

This code does the following:

1. Checks to see if the data set is SMS-managed by checking if a &SC (or &STORCLAS) is present.
2. Checks to see if the data set is non-SMS-managed production.
3. If the data set is not a production one, it will not be recovered from the out of space condition.

VSAM EOV with Media Manager Data Sets

Media Manager Services (MMS) is the latest generation of IBM access methods. Until recently, MMS was primarily used with DB2 Table Spaces, which are a special type of linear data set (LSDS). When SMS-managed multi-striped Extended Format (EF) data sets were introduced in OS/390 Release 2.10, they used MMS instead of the traditional Virtual Storage Access Method (VSAM) used with all other ESDS, KSDS, RRDS, non-SMS or SMS-managed, and single striped SMS-managed EF data sets. Starting with z/OS 1.3, all SMS-managed ESDS, KSDS, RRDS, LSDS, and striped EF data sets will use MMS. Non-SMS ESDS, KSDS, RRDS, and non-DB2 LSDS data sets will continue to use the VSAM access method.

There are some fundamental differences between the MMS and VSAM access methods. VSAM waits until the extent is 100% full before checking if there is an available candidate volume associated with the data set. MMS makes this check when the extent is less than 100% full. BrightStor CA-Allocate's VSAM EOV support is triggered by these checks for candidate volumes, which means that there will be occasions when MMS does not determine the need to extend onto the candidate volume that MMS requested.

Another difference is that MMS gets the primary and secondary allocation amounts from a different control block than VSAM gets it from. This will not prevent changes from being made to the applicable space amounts in the EOV_VSAM Environment. However, MMS will start using the new value immediately. This means if MMS decides to defer spanning onto the additional candidate volume just provided by BrightStor CA-Allocate's VSAM EOV support, and the secondary amount was changed in EOV_VSAM, then subsequent extents on the current volume will be with the new size.

Restrictions

While adding additional candidate volumes to DB2 Table Spaces, BrightStor CA-Allocate does not alter their primary and/or secondary allocation amounts.

Depending on the DB2 version, there could be from 3 to 5 DB2 started tasks running. When coding in the VDSPROG for DB2 allocations it is best to use the &JOB variable for keying on the DB2 started task name. It is very important to understand that the DB2 started job task is the one doing the allocation not the batch job with the DB2 commands.

When having problems with DB2, before calling Customer Support, you should do a modify to the BrightStor CA-Allocate started task to get diagnostics.

```
/F VAMSTC,DIAGS=xxxxDBM1
```

xxxxDBM1 = is the DBM started task name. The ending suffix is usually DBM1, the prefix varies.

Once this command is issued all diagnostics will appear in the xxxxDBM1 started task system log.

Rerun the failing job and have ready the diagnostics from the DB2 started task as well as the complete failing job log.

To stop diagnostics issue:

```
/F VAMSTC,DIAGS=
```

EOV Support for VSAM Data Sets with Unused Candidate Volumes

The purpose of intercepting these allocations is to be able to provide the ability to alter the initial primary and/or secondary space allocation amounts that will be used when a VSAM data set extends onto an already existing candidate volume. Since the data set is already open, the space amounts used will be what is indicated in the applicable system control blocks associated with the opened data set. When a data set is opened, the values in these control blocks contain the initial creation information.

If an EOV condition is encountered after the data set is opened, BrightStor CA-Allocate was used to change these space amounts, for example, during an EXTEND Environment invocation, then these values will reflect the new space amounts. When EOV_VSAM is launched, the &PRIMARYD/I and &SECONDARYD/I variables contain the originally defined VSAM data set information. These values may not be the same as the current space amounts contained in the applicable system control blocks.

The current values of the space amounts associated with the failing VSAM data or index component are available in the CURRENT_PRIMARYD/I and CURRENT_SECONDARYD/I variables. These four CURRENT_* variables are only applicable in the EOV_VSAM Environment, for all VSAM data sets, regardless of whether or not they have available candidate volumes, or whether or not they are SMS-managed. This means that for a failing VSAM component, there are four values to choose from for adjusting the space allocation amounts:

1. The value the data set was originally defined with, contained in the applicable PRIMARYD/I and SECONDARY/I variables.
2. The new value the data set has been given since it was opened, contained in the applicable CURRENT_PRIMARYD/I and CURRENT_SECONDARY/I variables.
3. The maximum available extent size in the storage group, retrieved by the LSPACE Support into the applicable LARGEST_EXT* variable.
4. Whatever amount a customer wants to set it to (up to the current IBM-documented maximum).

An example of (1) is shown below in Figure 2-12, where the current values are reinitialized to the original ones by "setting" the applicable &variable to itself (as in a 'SET &PRIMARYD = &PRIMARYD' statement).

FIGURE 2-12 Sample ASR for VSAM EOV With Candidate Volumes

```
IF &VAMENVIR = 'EOV_VSAM'                                   /* VSAM END-OF-VOLUME */
THEN
DO
        IF &CANDIDATE_VOLUMES = 'Y'                         /* AVAILABLE CANDIDATE VOLUMES */
        THEN
        DO
                IF &ABENDCOMP = 'DATA'                      /* EOV ON DATA COMPONENT */
                THEN
                DO
                        SET &PRIMARYD = &PRIMARYD           /* RESET TO ORIGINAL VALUE */
                        SET &SECONDARYD = &SECONDARYD       /* RESET TO ORIGINAL VALUE */
                END
                IF &ABENDCOMP = 'INDEX'                     /* EOV ON INDEX COMPONENT */
                THEN
                DO
                        SET &PRIMARYI = &PRIMARYI           /* RESET TO ORIGINAL VALUE */
                        SET &SECONDARYI = &SECONDARYI       /* RESET TO ORIGINAL VALUE */
                END
        END
END
```

EOV_VSAM Data Sets with Additional Volume Amount=Secondary

End-of-Volume Support for VSAM data sets with the Additional Volume Amount=SECONDARY data class attribute requires no special Allocation Selection Routine (ASR). IBM services updates the primary allocation amount in the pertinent MVS control blocks with the applicable secondary allocation amount from the cluster's catalog entry prior to checking for an available candidate volume for the data set to extend onto. VSAM_EOV is still using what is indicated in the primary allocation amount (that it finds in the applicable MVS control blocks) to make the first extension to an additional volume. Additional information can be found in the variable descriptions of **PRIMARYD** and **PRIMARYI** in the chapter "**Implementation**."

EOV Support for Zero Secondary

NonVSAM data sets allocated with no secondary space fall victim to D37-04 abends when their primary space is exhausted. Running with a **PLSZSEC** (Y) configuration will temporarily change the '0' secondary allocation amount to '1' which will result in the EXTEND Environment being launched just as if it would have been launched had the data set originally been allocated with a non-zero secondary.

EXTEND Support for these 'Z SEC' allocations will be identical to what is available to original 'NZ SEC' allocations. This includes changing the secondary value or even denying the extend request by exiting the environment with something other than the default return code of (0). They can be identified by the EXTEND Environment ASR by querying the value of the **SECONDARY_HAD_ZERO** variable.

LSPACE Support for Primary and Secondary Allocation

LSPACE support allows the Storage Administrator the opportunity to reduce primary and / or secondary space allocations to avoid X-37 type abends. This LSPACE function calculates the largest extents (1-5, depending on current number of extents) in one of the following allocation types:

- Cylinders
- Megabytes
- Tracks

This enables the ASR writer to determine whether or not the current allocation fits on the target volume.

In order to support this function, several ASR variables are available within the ALLOC, DEFINE, EOV, EOV_VSAM, and EXTEND environments. The following is a list of these variables.

TABLE 2-5 List of LSPACE Variables

Variable Name	Alias
&LARGEST_EXTCYL	&LEC
&LARGEST_EXTCYL_NOT	&LECN
&LSPACE_PCTAFT_CYLNOT	&LPACN
&LSPACE_PCTAFT_MB	&LPAM
&LARGEST_EXTMB	&LEMB
&LARGEST_EXTMB_NOT	&LEMBN
&LARGEST_EXTTRK	&LET
&LARGEST_EXTTRK_NOT	&LETN
&LSPACE_FREESPACE	&LFF
&LSPACE_PCTAFT_CYL	&LPAC
&LSPACE_PCTAFT_MBNOT	&LPAMN
&LSPACE_PCTAFT_TRK	&LPAT
&LSPACE_PCTAFT_TRKNOT	&LPATN
&LSPACE_RETURN_CODE	&LRC

Variable Name	Alias
&LSPACE_STORGRP	&LS
&LSPACE_VOLUME	&LV

Note: LSPACE processing is available at the storage group or volume level. The following ASR statements are illustrations of each:

```
SET &LV = 'vvvvvv' /* <<== volser */
SET &LS = 'ssssss' /* <<== storage group */
```

Options

There are many options available within the LSPACE function, and depending upon which BrightStor CA-Allocate environment is active, the Storage Administrator should consider the following before implementing this support:

- In the ACS environment, only the &LSPACE_VOLUME variable is allowed because the only LSPACE support available during ACS is to decide whether to remove the “Guaranteed Space” attribute, and this depends on the space available on the current volume, not in the storage group as a whole.
- In the ALLOC and DEFINE environments, either the &LSPACE_VOLUME or the &LSPACE_STORGRP variables are allowed. The &LSPACE_VOLUME variable should be used for either SMS-managed allocations specifying the “Guaranteed Space” attribute, or for non-SMS-managed specific volume requests that BrightStor CA-Allocate are not redirecting. The &LSPACE_STORGRP variable should be used whenever BrightStor CA-Allocate is planning to redirect a Non-SMS-managed allocation or when an SMS-managed allocation has not been requested with the “Guaranteed Space” attribute.

  Note: To intercept SMS-managed data set allocations in the ALLOC and DEFINE environments, “PLSOPT10 (Y)” needs to be specified in the VKGPARMS member of PARMLIB.
- In the EOV and EOV_VSAM environments, only the &LSPACE_STORGRP variable can be used because it is too late for further processing on the current volume and the new volume is not yet known.
- In the EXTEND environment, only the &LSPACE_VOLUME variable is allowed because volume selection can not be changed.
- After the &LSPACE_VOLUME or &LSPACE_STORGRP variables are set and one of the other variables is queried, the actual LSPACE process is performed. The &LFF variable is a flag that is set when the Freespace Threshold is violated. It only applies if a storage group (not a volume) is being processed.

- After the LSPACE process is complete, if &LSPACE_RETURN_CODE contains a non-zero return code, some or all of the other variables may be invalid. If &LSPACE_RETURN_CODE is zero, all of the other variables were set to valid values. For more information, see the variable description for "**LSPACE_RETURN_CODE**" in the chapter "**Implementation**."
- The contents of &LEC, &LEMB, and &LET is the largest possible extents in cylinders, megabytes, and tracks respectively. What is the difference? Assume a storage group has two volumes, a 3390 and a 3380. Also, assume that the 3390 has 140 available tracks, and a 3380 has 150 available tracks. The contents of &LET is 150 which is 6MB of data (7,121,400 bytes = 150 tracks * 47,476 bytes/track), but &LEMB contains 7 (7,932,960 bytes = 140 tracks * 56,664 bytes/track). &LPAC, &LPAM, and &LPAT contains the freespace percent after the allocation, assuming the &LEC, &LEMB, and &LET values were used respectively.
- &LFF is set to `Y' when any of the &LEC, &LEMB, and &LET variables violate the freespace threshold &LECN, &LEMBN, and &LETN contains the largest possible extents in cylinders, megabytes, and tracks that would not violate the freespace threshold. &LPACN, &LPAMN, and &LPATN contains the freespace percent after the allocation.
- &LFF, &LPACN, &LPAMN, and &LPATN variables only have meaning for non-SMS-managed allocations.

How Other Variables Can Affect the Process

Some variables can alter the LSPACE process. Below is a list of variables and the role they play:

TABLE 2-6 How Other Variables Affect LSPACE

Variable	How it affects the process
&UNIT	BrightStor CA-Allocate only picks a volume from a storage group if that volume belongs to &UNIT. For example, assume a job allocates a data set with UNIT=3380. Also assume that MIXED is a BrightStor CA-Allocate storage group with a mixture of 3380 and 3390 devices. BrightStor CA-Allocate only chooses 3380 devices from the storage group because 3390 devices do not belong to the unit 3380. Therefore, the LSPACE process only examines volumes that reside in both the &STORGRP and &UNIT.
&GSA	"Guaranteed Space" is only applicable to specific volume requests for SMS-managed allocations associated with a Storage Class specifying this attribute. The SMS volume selection routines fail such allocations when the target volume does not have sufficient freespace. BrightStor CA-Allocate offers two choices in such cases: 1. Change the Storage Class to one without "Guaranteed Space." 2. Reduce the primary allocation amount to the largest available amount retrieved by the LSPACE LV process. Storage Class changes must be done in the ACS Environment. Changing the primary allocation amount must be done in either the ALLOC or DEFINE environment.
&CONTIG	Under most circumstances, DADSM breaks an extent request into as many as five extents. Therefore, the LSPACE process sums the five largest extents. However, if the allocation requires contiguous space, only the single largest extent is considered. This is applicable to Non-VSAM only.
&EXTENTS	Under most circumstances, DADSM breaks an extent request into as many as five extents. Non-VSAM data sets can only have 16 extents to a volume. If an EXTEND of a Non-VSAM data set is being processed and the data set is already in more than 11 extents, then DADSM cannot supply this secondary allocation in 5 extents. Therefore, if extents is greater than 11, then the N largest extents is used where N = 16-&EXTENTS.

Variable	How it affects the process
&POOLSUB	In the volume selection environments (that is, ALLOC, DEFINE, EOV, and EOV_VSAM) the &POOLSUB variable allows an automatic storage group selection process. Since many sites use this feature, it may not be efficient to SET &LSPACE_STORGRP to a storage group name. Therefore, after &POOLSUB has been set to 'Y', then &LSPACE_STORGRP is set to the first storage group that &POOLSUB finds. See the variable description for "**POOLSUB**", in the chapter "**Implementation**", for more information.

Special Considerations

- LSPACE processing can be thought of as a picture of a volume at a particular point in time. There is NO guarantee that the free space available at the time the LSPACE was issued is available for allocation.
- However, the time frame from when BrightStor CA-Allocate does the LSPACE and the time the allocation is actually made is relatively small, so the space is likely to be available.
- LSPACE processing requires I/O and causes additional overhead when being used. When doing LSPACE processing for a storage group, all VTOC's within the storage group must be read before any variables can be returned to the ASR. As the number of volumes within the storage group grows, so does the overhead involved in obtaining the required information.
- VSAM space allocations are sometimes rounded up from the initial request. One example is when the request is for a KSDS data set and space is requested for the data component but not the index. To prevent some allocations from failing, the ASR writer might not want to use the entire &LEC, &LET, OR &LEMB value.

The following figures (Figure 2-13, Figure 2-14, and Figure 2-15) have sample ASR code showing the LSPACE support:

FIGURE 2-13 Sample LSPACE ASR, Part 1

```
Menu     Utilities    Compilers    Help
------------------------------------------------------------------------
BROWSE          .ASR.PARMLIB(LSPACE#1) - 01.06          Line 00000000 Col 001 080
Command ==> _____________________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************
SET &UNIT = 'SYSALLDA'                  /* PREPARE FOR LSPACE PROCESSING  */
IF &VAMENVIR = 'EXTEND'
   THEN SET &LV = &LV                   /*   NEED CURRENT VOLUME TO ...   */
IF &VAMENVIR = EOV*                     /*  VAMENVIR = EOV OR EOV_VSAM    */
   THEN SET &POOLSUB = 'Y'              /* USE CURRENT STORGE GROUP TO .. */
IF &VAMENVIR = 'ACS' AND &GSA = 'Y'     /* GUARANTEED SPACE MUST USE THE  */
   THEN SET &LV = &ALLVOL               /* TARGET VOLUME FOR LSPACE...    */
IF &VAMENVIR = 'ALLOC' OR &VAMENVIR = 'DEFINE' THEN DO
   IF  &GSA = 'Y' THEN                  /*   GUARANTEED SPACE NEEDS...    */
       SET &LV = &ALLVOL                /*       LSPACE LV PROCESSING     */
   ELSE DO                              /*    OTHERWISE WILL NEED...      */
       IF &SC ¬= ''  THEN               /*          FOR SMS-MANAGED...    */
          SET &LS = &SG                 /* NEED THE DFSMS STORAGE GROUP   */
       ELSE SET &LS = 'DASD80'          /*   NONSMS NEEDS THE ALLOCATE SG */
        END
                                            END
COPYBOOK 'LSPACE#2'                    /* ... TO INVOKE LSPACE PROCESSING   */
******************************** Bottom of  Data ********************************
```

FIGURE 2-14. Sample LSPACE ASR, Part 2

```
Menu     Utilities    Compilers    Help
------------------------------------------------------------------------
BROWSE          .ASR.PARMLIB(LSPACE#2) - 01.04          Line 00000000 Col 001 080
Command ==> _____________________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************
IF &LRC = 0 THEN             /* INVOCATION OF THE LSPACE PROCESS           */
   DO
     IF &SPACETYPE     = 'CYL' OR     /*        CYLINDERS?                 */
        &SPACETYPED    = 'CYL' OR &SPACETYPEI  = 'CYL'
        THEN
        DO
          IF &VAMENVIR = 'EXTEND' OR &VAMENVIR = 'EOV'
               THEN   /* INCREASE TO REDUCE DATA SET FRAGMENTATION OR ... */
               DO     /* REDUCE TO PREVENT SPACE-NOT-AVAILABLE FAILURE    */
                      IF &SECONDARY  > 0 THEN SET &SECONDARY  = &LEC
                      IF &SECONDARYD > 0 THEN SET &SECONDARYD = &LEC
                      IF &SECONDARYI > 0 THEN SET &SECONDARYI = &LEC
               END
          COPYBOOK 'LSPACE#3'      /* FOR ACS, ALLOC, DEFINE, EOV_VSAM */
        END
   END
******************************** Bottom of  Data ********************************
```

FIGURE 2-15 Sample LSPACE ASR, Part 3

```
Menu     Utilities    Compilers    Help
------------------------------------------------------------------------
BROWSE          .ASR.PARMLIB(LSPACE#3) - 01.04          Line 00000000 Col 001 080
Command ==> _____________________________________________________ Scroll ==> CSR

********************************* Top of  Data ********************************
IF &VAMENVIR = 'ACS' OR &VAMENVIR = 'ALLOC' OR &VAMENVIR = 'DEFINE'
   OR &VAMENVIR = 'EOV_VSAM' THEN DO
      IF &VAMENVIR = 'ACS' AND &GSA = 'Y'    /* FOR GUARANTEED SPACE...   */
        AND &HLQ = PROD.**  THEN          /* ...WITH THESE DATA SETS      */
   /* ...SWITCH TO A SC WITHOUT IT WHEN THERE IS INSUFFICIENT SPACE      */
        DO
          IF &PRIMARY  > 0 AND &LEC < &PRIMARY  THEN SET &SC = 'NOTGSA'
          IF &PRIMARYD > 0 AND &LEC < &PRIMARYD THEN SET &SC = 'NOTGSA'
          IF &PRIMARYI > 0 AND &LEC < &PRIMARYI THEN SET &SC = 'NOTGSA'
        END
      ELSE      /* INCREASE PRIMARY TO REDUCE DATA SET FRAGMENTATION OR   */
      DO        /* REDUCE IT TO PREVENT A SPACE-NOT-AVAILABLE FAILURE     */
                     IF &PRIMARY  > 0 THEN SET &PRIMARY  = &LEC
                     IF &PRIMARYD > 0 THEN SET &PRIMARYD = &LEC
                     IF &PRIMARYI > 0 THEN SET &PRIMARYI = &LEC
      END
   END
******************************** Bottom of  Data ******************************
```

NOT CATLGD 2 Support

Allocating a new Non-VSAM data set using the same name as an already cataloged data set results in a NOT CATLGD 2 condition. This means that the data set was successfully allocated, but the catalog action failed.

BrightStor CA-Allocate provides 3 variables to avoid this condition:

- "**STOP_NOT_CATLG2**"
- "**STOP_NOT_CATLG2_NODE**"
- "**STOP_NOT_CATLG2_RC**"

These variables allow the user to either rename, uncatalog, or delete the existing data set when the condition is detected in the SPACE environment.

Restrictions

The following restrictions should be considered before using this support:

1. New data sets that are allocated from a TSO session are not supported.
2. The UNCATALOG option cannot be performed against an existing SMS-MANAGED data set, because SMS does not permit uncataloged data sets.
3. Existing data sets that span more than 20 volumes are not supported.
4. To be eligible for this support, non-SMS-managed non-VSAM data sets must have been redirected by BrightStor CA-Allocate (that is, a STORGRP must be set in the ALLOC environment).
5. Existing data sets cataloged to a pseudo-volser are ignored.
6. The UNCATALOG option is not performed if the existing cataloged data set is on the same volume that the new data set is being allocated to.
7. The DELETE option can not be performed on a data set referenced by a SUBSYS DD statement, because SUBSYS data sets are considered SYSIN / SYSOUT data sets. IBM prevents these data sets from getting deleted.

Sample ASR Statements

The following figures (Figure 2-16 and Figure 2-17) show sample ASR code using the NOT CATLGD 2 support:

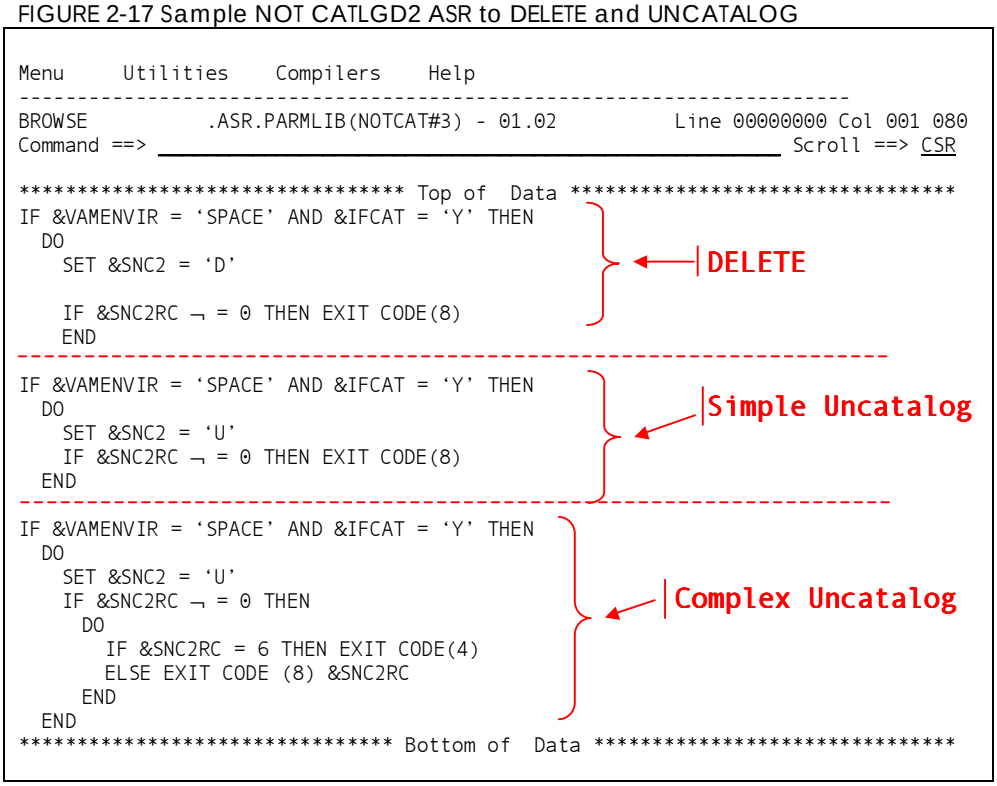
FIGURE 2-16 Sample NOT CATLGD2 ASR to RENAME (Simple and Complex)

```
Menu     Utilities    Compilers    Help
------------------------------------------------------------------------
BROWSE          .ASR.PARMLIB(NOTCAT#1) - 01.02          Line 00000000 Col 001 080
Command ==> _____________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************
IF &VAMENVIR = 'SPACE' AND &IFCAT = 'Y' THEN DO
   SET &SNC2 = 'R'
   SET &SNC2N = 'NOTCAT'

   IF &SNC2RC ¬ = 0 THEN EXIT CODE(8)
                                          END
------------------------------------------------------------------------
IF &VAMENVIR = 'SPACE' AND &IFCAT = 'Y' THEN DO
   SET &SNC2 = 'R'
   SET &SNC2N = 'NOTCAT'
   IF &SNC2RC ¬ = 0 THEN DO
      IF &SNC2RC = 11 THEN DO
           SET &SNC2N = 'NC2'
           IF &SNC2RC = 11 THEN EXIT CODE(8)
                           END
      IF &SNC2RC = 12 THEN DO
           SET &SNC2N = 'NC2'
           IF &SNC2RC = 12 THEN EXIT CODE(8)
                           END
      IF &SNC2RC ¬ = 0 THEN EXIT CODE(8)
                        END
                                          END
******************************** Bottom of  Data ********************************
```

Simple

Complex

FIGURE 2-17 Sample NOT CATLGD2 ASR to DELETE and UNCATALOG



ACS Support

The ACS environment can be used as a direct replacement for IBM ACS routines. Because BrightStor CA-Allocate's ASR language can be used to manage both SMS-managed and non-SMS-managed data sets, there is no need for ACS routines or ACS exits.

Note: IBM procedures may require some minimum ACS Routine setup. Refer to the appropriate IBM documentation.

Any of the four SMS constructs can be changed: DATA CLASS, STORAGE CLASS, MANAGEMENT CLASS, or STORAGE GROUP.

Normal ISMF setup is done to create and define the constructs. BrightStor CA-Allocate gets control after all IBM ACS processing is complete, but before returning to the original requestor of the ACS services. It can be used to change or remove any of the four SMS construct names.

How to Use the ACS Support

The ACS environment is entered anytime ACS processing is required for any of the four SMS constructs. The variables available are:

- DATACLAS — DC or DATACLASS
- STORCLAS — SC or STORAGECLASS
- MGMTCLAS — MC or MANAGEMENTCLASS
- STORGRP — SG or STORAGEGROUP

Storage group can be defined for either SMS or BrightStor CA-Allocate. If a Storage Class is set, the data set becomes SMS-managed, and the STORGRP must be a valid SMS Storage Group name. If the Storage Class is NOT set, the data set becomes non-SMS-managed, and the STORGRP must be a valid BrightStor CA-Allocate Storage Group.

Enhanced DATACLAS Support

This feature relieves the SMS limitation that values specified for the DATACLAS sub-parameters is only used if others are not explicitly coded in the JCL. At this time, this support is limited to new data set allocations intercepted in the ALLOC and DEFINE Environments. As delivered, BrightStor CA-Allocate automatically exempts new SMS-managed data set allocations from ALLOC and DEFINE Environments processing.

For implementation procedures, see “**PLSOPT10**” in the chapter “**Implementation**.”

IBM's DATACLAS Support

The SMS Data Class parameter (DATACLAS) can be set by either JCL or the IBM ACS routines. The sub-parameters associated with a DATACLAS (that is, RECORG, LRECL, SPACE, RETPD, and so on) are propagated to the data set unless that same parameter is specified in the JCL. The JCL information overrides what was defined in the DATACLAS.

The problem with this is two fold:

1. If a change is needed to a particular parameter in many jobs, then the JCL may have to be changed everywhere because the JCL overrides the ACS DATACLAS.
2. Any user can override DATACLAS specifications coded by the installation in the IBM ACS routines by specifying their own DATACLAS information in the JCL.

DATACLAS Support

In the ALLOC Environment, the ASR has the ability to interrogate both the DATACLAS and JCL information and choose from either one, or specify an entirely different value. The JCL information is what is used to allocate the data set if BrightStor CA-Allocate does the volume selection. This means that any DATACLAS information needs to be propagated to the related JCL parameter if a STORGRP is set for a non-SMS-managed allocation.

The sample Allocation Selection Routine shown below in Figure 2-18 shows this. Note that two different types of processing are done for allocations with an associated DATACLAS. For the 'ISPDATA1' Data Class, the ASR writer knows which sub-parameters have values defined, so there is no need to check the associated DC_*_FLAG indicator. In addition, all allocations with this Data Class are to be converted to the same average records type and quantity. For any other Data Class, what action the ASR takes depends on what sub-parameters have been defined, which is determined by querying the associated DC_*_FLAG indicator.

Figure 2-18 has ASR code addressing VSAM JCL DEFINEs that are intercepted in the ALLOC Environment. Note the use of 'EXIT CODE(0)'. VSAM JCL DEFINEs are intercepted first in the ALLOC Environment and then in the DEFINE Environment. Redirection needs to be done in the DEFINE Environment.

FIGURE 2-18 Sample Enhanced DATACLAS ASR Part 1

```
Menu     Utilities    Compilers    Help
------------------------------------------------------------------------
BROWSE          .ASR.PARMLIB(ENHANDC) - 01.04          Line 00000000 Col 001 080
Command ==> _____________________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************
FILTLIST &VSAM INCLUDE('DCKSDS','DCESDS','DCLSDS','DCRRDS')
IF &VAMENVIR = 'ALLOC' THEN
   DO
     IF  &DC = '' THEN EXIT CODE (0)         /* IF NONE THEN EXIT      */
     IF  &DC = &VSAM THEN EXIT CODE (0)      /* VSAM DATACLAS EXIT     */
     IF  &DSORG = 'VS' THEN EXIT CODE (0)    /* VSAM JCL DEFINE EXIT   */
     IF  &DC_RECFM = '' THEN EXIT CODE (0)   /* VSAM JCL DEFINE EXIT   */
     IF  &DC = 'ISPDATA1' THEN               /* FOR THIS DATA CLASS    */
         DO
            COPYBOOK 'DC1'                   /* ENFORCE DC ATTRIBUTES  */
        END
     ELSE
         DO
            COPYBOOK 'DC2'                   /* ALL OTHER DATA CLASSES */
        END
   END
IF &VAMENVIR = SPACE AND &BLKSIZE = 0 THEN SET &BLKSIZE = &OPTBLK
****************************** Bottom of  Data ******************************
```

Sample Enhanced DATACLAS ASR, Part 2: COPYBOOK DC1

```
SET &SPACCVT = 'N'                           /* DON'T DO SPACE CONVERSION  */
SET &SPACTYPE = 'AVR'                        /* SPACE TYPE AVERAGE RECORDS */
SET &AVGRECTYPE = &DC_AVGRECTYPE             /* USE DATACLAS ATTRIBUTE     */
SET &AVGRECSIZE = &DC_LRECL                  /* SET AVRSIZE                */
SET &PRIMARY = &DC_PRIMARY                   /* USE DATACLAS ATTRIBUTE     */
SET &SECONDARY = &DC_SECONDARY               /* USE DATACLAS ATTRIBUTE     */
SET &RETPD = &DC_RETPD                       /* USE DATACLAS ATTRIBUTE     */
SET &LRECL = &DC_LRECL                       /* USE DATACLAS ATTRIBUTE     */
SET &UNIT = 'SYSALLDA'                       /* SET UNIT                   */
SET &STORGRP = 'SSLDA'                       /* SET STORGRP                */
```

Sample Enhanced DATACLAS ASR, Part 3: COPYBOOK DC2

```
                                          /* IF AVGBLK SPECIFIED        */
                                          /* THEN USE IT                */
IF &DC_AVGBLK_FLAG = 'Y'
THEN
DO
       SET &SPACCVT = 'N'                 /* DO NOT DO SPACE CONV       */
       SET &SPACTYPE = 'BLK'              /* SPACE TYPE BLOCKS          */
       SET &AVGBLK = &DC_AVGBLK           /* USE DATACLAS ATTRIBUTE     */
       SET &BLKSIZE = &DC_AVGBLK          /* USE DATACLAS ATTRIBUTE     */
END
IF &DC_AVGRECTYPE)FLAG = 'Y' AND &DC_AVGRECSIZE_FLAG = 'Y'
THEN
DO                                        /* DO NOT DO SPACE CONV       */
       SET &SPACCVT = 'N'
       SET &SPACTYPE = 'AVR'
       SET &AVGRECTYPE = &DC_AVGRECTYPE   /* USE DATACLAS ATTRIBUTE     */
       SET AVGRECSIZE = &DC_AVGRECSIZE    /* USE DATACLAS ATTRIBUTE     */
END
                                          /* ENFORCE DATACLAS ATTRIBUTES */
IF &DC_PRIMARY_FLAG = 'Y' THEN SET &PRIMARY = &DC_PRIMARY
IF &DC_SECONDARY_FLAG = 'Y' THEN SET &SECONDARY = &DC_SECONDARY
IF &DC_RETPD_FLAG = 'Y' THEN SET &RETPD = &DC_RETPD
IF &LRECL = 0
THEN
DO
       IF &DC_LRECL_FLAG = 'Y' THEN SET &LRECL = &DC_LRECL
       ELSE SET &LRECL = 80
END
SET &UNIT = 'SYSALLDA'
SET &STORGRP = 'POOLB'
```

Tape Allocation Support

Tape Allocation Support provides the ability to enforce standards for new tape allocations. For example, data compression may be enforced, Label Types adhered to, and migration to new tape devices accomplished without the need for end users to make JCL changes. Also, available disk space shortages may be avoided by redirecting extremely large disk allocations to tape, and tape utilization may be made more efficient by redirecting identifiable small tape allocations to disk.

Technical Overview

BrightStor CA-Allocate intercepts allocation requests for new tape data sets in the ALLOC Environment. Three tape-only variables exist to support tape allocations:

- &CMP3480 (data compression)
- &FILENUM (file sequence number)
- &LABEL (label type)

All three variables are modifiable. Some internal validity checks are performed. For example, &LABEL can be changed to only one of the eight valid types specified in any MVS JCL Reference Manual, &FILENUM to only a positive integer, and &CMP3480 to only “on” or “off.” Detailed information can be found in **TABLE 4-3 List of Variables** in the chapter “**Implementation**.”

Even with the internal validation, carefully consider before modifying these variables. For example, globally forcing data compression “on” for all tapes is not advisable if tapes may be shipped to other sites that do not support the compression feature on their 3480/3490 tape devices. Neither is it a good idea to change the label type to one that, although a valid MVS one, is unsupported at your particular installation. Finally, an incorrect value for &FILENUM can cause various errors, including unintentionally over writing an existing data set or failing with a 413 or A13 abend.

Device-Type Conversions

BrightStor CA-Allocate also supports the following device-type conversions:

- TAPE- t o- TAPE
- TAPE-to-DISK
- DISK-to-TAPE

TAPE-to-DISK and DISK-to-TAPE conversions are requested by making the appropriate change to the &UNIT variable in the ALLOC Environment. Before ALLOC, the incoming device class is determined and saved. An internal check after ALLOC for a device class change determines, depending on the conversion type, whether to take a certain series of actions. These actions are discussed in detail later in this section.

Additionally, four variables exist to detect and disable volume referback or unit affinity specifications that may adversely impact device-type conversions:

- &UNITAFF (detect / disable UNIT=AFF=ddname)
- &VREFDDN (detect / disable VOL=REF=*.ddname)
- &VREFDSN (detect / disable VOL=REF=dsname)
- &VREFSTP (detect / disable VOL=REF=*.stepname.ddname)

These four variables are as applicable to tape-only and disk-only allocations as they are to device-type conversion operations. They can be displayed and modified in the ALLOC Environment. More information is located in **TABLE 4-3 List of Variables** in the chapter "**Implementation**."

Not all data sets are good candidates for device type conversions. Program products that use internal parameters to control output device characteristics are particularly poor candidates. If the device type is changed for new data sets created by DFHSM, SMS, or BrightStor CA-Disk, the results can be unpredictable.

TAPE-to-TAPE Conversions

TAPE-to-TAPE conversions occur when a request is issued to change the tape unit type specified for the original allocation to another tape unit type. If no additional changes are explicitly requested for the data compression, file sequence number, label type, and expiration date (or retention period) variables, what was previously specified for the original allocation is used.

Nothing is done to any volume referback or unit affinity that may have been originally specified for the allocation. These specifications are typically not an impediment in TAPE-to-TAPE conversions because the original and new device types are within the same device class.

The following sample ASR statements result in a TAPE-to-TAPE device-type conversion from round tapes to cartridges with data compression being “turned on” for the allocations unless TRTCH=NOCOMP has been explicitly specified in the JCL.

```
IF &UNIT = '3420' THEN
   DO
       SET &UNIT = '3480'
       IF &CMP3480 EQ 'N'            /* NO COMPRESSION ASKED FOR?     */
       THEN SET &CMP3480 = 'Y'       /* IF NOT, TURN IT ON            */
   END
```

TAPE-to-DISK Conversions

TAPE-to-DISK conversions occur when a request is issued to change what was originally a tape allocation to disk. If no additional changes are requested for the expiration date (or retention period) variable, what was specified for the original allocation is used. Data compression is turned off and the file sequence number and label type is reset to 1 and SL, respectively. Allocation quantity has no default. BLK, CYL, or TRK can be specified as the allocation unit and failure to do so generally leads to message IEF127I. If BrightStor CA-Allocate was requested to redirect the allocation but is unable to select a volume, it fails the allocation through the standard &FINVS='Y' route.

Any volume referback or unit affinity originally specified for the allocation is disabled. These specifications typically have an adverse impact on TAPE-to-DISK conversions because the original and new device types are of a significantly different device class. If either is found, and BrightStor CA-Allocate is not requested to redirect the allocation, the conversion request is ignored.

The following sample ASR statements result in a TAPE-to-DISK device-type conversion. In the event that no space attributes had been specified for the original allocation, the default of SPACE=(CYL,(10,10),RLSE) is used.

```
IF &UNIT = 'TAPE' THEN
   DO
   SET &UNIT = 'SYSALLDA'
   SET &STORGRP = 'DASD80'
   IF &SPACTYP = '' THEN
       DO
       SET &SPACTYPE = 'CYL'
       SET &PRIMARY = 10
       SET &SECONDARY = 10
       SET &RLSE = 'Y'
       END
   END
```

DISK-to-TAPE Conversions

DISK-to-TAPE conversions occur when a request is issued to change what was originally a disk allocation to a tape allocation. Any allocation unit and quantities that were previously specified for the original allocation are ignored. Unless new values are requested, the file sequence number and label type remain the same as they were either specified or implied in the original disk allocation (typically 1 and SL). No default data compression is assumed. Any expiration date (or retention period) that was specified for what was originally a DASD allocation is still used, unless a new date is requested. And lastly, the newly converted tape allocation is nonspecific.

Any volume referback originally specified for the allocation is disabled. These specifications typically have an adverse impact on DISK-to-TAPE conversions because the original and new device types are of a different device class.

The following sample ASR statements result in a DISK-to-TAPE device-type conversion:

```
IF &UNIT = 'SYSDA' THEN
   DO
       SET &UNIT = '3480'
   END
```

Volume Referbacks/Unit Affinity and Disk-only Allocations

When redirection is requested, any volume referback or unit affinity that may have been originally specified for the allocation should be disabled. This is necessary to ensure the successful allocation on the volume chosen. Failure to do so causes '**IEF245I INCONSISTENT UNIT NAME AND VOLUME SERIAL**' and the allocation fails. To avoid allocation failures, requests to disable volume referback or unit affinity are ignored unless redirection is requested.

Note: You can automatically disable unit affinity by specifying PLSOPT3 with a value of 'Y'. For details, see "PLSOPT3" in the chapter "Implementation."

The following sample ASR statements illustrate some of the flexibility allowed in this area. Volume referback and unit affinity are permitted for temporary data sets and tape, while permanent disk data sets are redirected, with volume referback and unit affinity disabled to prevent allocation failures.

FIGURE 2-19 Sample Referbacks and Unit Affinity ASR Statements

```
Menu     Utilities    Compilers    Help
------------------------------------------------------------------------
BROWSE          .ASR.PARMLIB(REFERBKS) - 01.03          Line 00000000 Col 001 080
Command ==> _____________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************
FILTLIST &DASD INCLUDE('SYSDA','3380','3390','DISK')
IF &UNIT = &DASD THEN
   DO
     IF  (&UNITAFF = 'Y') OR (&VREFDDN = 'Y') OR
     IF  (&VREFDSN = 'Y') OR (&VREFSTP = 'Y') THEN
         DO
            IF &DSTYPE = 'TEMP' THEN EXIT CODE(0)
            SET &UNITAFF = 'N'
            SET &VREFDSN = 'N'
            SET &VREFSTP = 'N'
            SET &VREFDDN = 'N'
            SET &UNIT = 'SYSALLDA'
            SET &STORGRP = 'POOL01','POOL02','POOL03'
        END
   END
******************************** Bottom of  Data ********************************
```

Understanding How Volume Referbacks and Unit Affinities Are Resolved

DDs are processed one at a time. Volume referbacks and unit affinities are resolved by the operating system immediately before its processing of a DD. This resolution removes any evidence of a device-type conversion.

Consider the following JCL:

```
File   Edit   Confirm   Menu   Utilities   Compilers   Test   Help
------------------------------------------------------------------------
EDIT            .ASR.PARMLIB(IEBCOPY) - 01.00                Columns 00001 00072
Command ==> ____________________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************
000001  //JOB        JOB
000002  //UNLOAD2    EXEC PGM=IEBCOPY
000003  //IN1          DD DSN=DSET.PO1,DISP=SHR
000004  //IN2          DD DSN=DSET.PO2,DISP=SHR
000005  //IN3          DD DSN=DSET.PO3,DISP=SHR
000006  //OUT1         DD DSN=UNLOADED.PO1,UNIT=3480,DISP=(NEW,PASS),
000007  //           LABEL=(1,SL)
000008  //OUT2         DD DSN=UNLOADED.PO2,UNIT=AFF=OUT1,DISP=(NEW,PASS),
000009  //           LABEL=(2,SL),VOL=REF=*.OUT1
000010  //OUT3         DD DSN=UNLOADED.PO3,UNIT=AFF=OUT1,DISP=(NEW,PASS),
000011  //           LABEL=(3,SL),VOL=REF=*.OUT1
000012  //SYSUT3       DD UNIT=SYSDA,SPACE=(CYL,(2,2))
000013  //SYSUT4       DD UNIT=SYSDA,SPACE=(CYL,(2,2))
000014  //SYSPRINT     DD DUMMY
000015  //SYSIN        DD *
000016  COPY INDD=IN1,OUTDD=OUT1
000017  COPY INDD=IN2,OUTDD=OUT2
000018  COPY INDD=IN3,OUTDD=OUT3
000019  /*
```

As written, OUT1 refers to a tape volume. If the ALLOC ASR changes the unit to a disk device, this device-type conversion is detected and the documented default actions are taken. Volume referbacks and unit affinities to OUT1 are no longer to a tape volume and unit, but rather to the disk one on which the allocation was made.

As written, with both volume referbacks and unit affinities, OUT2-OUT3 refer to a tape volume and unit. Any changes made to the device type associated with OUT1, as well as any volume that becomes associated with OUT1, are propagated forward to OUT2-OUT3 as each volume referback and unit affinity is resolved by the operating system. Because this resolution occurs before the OUT2-OUT3 are processed, no device-type conversion are detected. These DDs are seen as disk allocations with volume referback and unit affinity specified, but no space. The unit affinity attribute are only disabled because it is not applicable to disk allocations. No default space attributes are set. This needs to be done by the ALLOC ASR. Failure to do so results in the job failing with a '**IEF127I NO SPACE PARAMETER**' JCL error.

Virtual Tape Systems (VTS)

To redirect to a Virtual Tape System (VTS) you need to have PTF SS05223 applied and SET a VTS unit in the ALLOC environment. If converting a non-VTS tape allocation to a VTS tape system, when VOL=REF= and / or UNIT=AFF= are coded, these attributes should be kept, not shut off as is common for TAPE to DISK conversions.

VTS redirection problems

Failure to a set a VTS unit in the ALLOC environment may generate these error messages:

```
IEF344I jobname step DD - ALLOCATION FAILED DUE TO DATA FACILITY SYSTEM ERROR
IGD17207I VOLUME SELECTION HAS FAILED - THERE ARE NO ACCESSIBLE VOLUMES FOR DATA SET
IGD17277I THERE ARE (0) CANDIDATE VOLUMES
```

DFSMShsm Support

To activate HSM support, you must do two things:

1. Specify PLSOPT15(Y)
2. Use HSMON command to activate

You must take both actions to activate HSM support.

The interface with DFSMShsm allows DFSMShsm-allocated data sets to be handled like all other data sets. There is no need to create a DFSMShsm RECALL exit. Volume pools are maintained in one central location (BrightStor CA-Allocate storage groups). This eliminates the need to keep DFSMShsm's “pools” synchronized with BrightStor CA-Allocate.

In addition to the usual BrightStor CA-Allocate ASR variables, special-purpose variables are provided for DFSMShsm RECALL and RECOVER operations. These variables provide additional information about the data set which DFSMShsm is operating on:

&HSMDSN	Real data set name
&HSMGROUP	Security group name of requesting user
&HSMHLQ	Real data set high-level qualifier
&HSMJOB	Requesting job name
&HSMLLQ	Real data set low-level qualifier

&HSMNQUAL Num qualifiers in real data set name

&HSMUSER Requesting user name

The following sections present the details of DFSMShsm operation as they pertain to BrightStor CA-Allocate.

Data Movers Used by DFSMShsm

DFSMShsm uses multiple data movers to move data around. DFSMShsm may move the data itself, or it may call DFSMSdss to move the data. The data mover used depends on the type of operation and the type of data. Please consult the appropriate IBM documentation for more information about this.

SMS Construct Names Used During DFSMShsm RECALL and RECOVER Operations

During the processing of a DFSMShsm RECALL or RECOVER which is not SMS-managed, BrightStor CA-Allocate uses special SMS construct names. These values indicate to BrightStor CA-Allocate operating system interface programs that the non-SMS-managed DFSMShsm RECALL or RECOVER is being managed by BrightStor CA-Allocate.

Table 2-7. SMS Construct Names Used

SMS Construct	Value
Storage Class	SAMSALSC
Storage Group	SAMSALSG

Do **not** define these SMS constructs names to your SMS configuration.

If you wish, you can customize these names by setting the PLSSC and PLSSG parameter values to the other unique Storage Class and Storage Group names. The default values are as indicated in Table 2-7.

New Logic Required in the ACS Environment

In order for BrightStor CA-Allocate to operate on allocations performed by DFSMShsm during RECALL and RECOVER, the &STORCLAS variable must be set to the PLSSC parameter value in the ACS Environment. This tells BrightStor CA-Allocate to "manage" the allocation. Not doing so allows the allocation to proceed as if BrightStor CA-Allocate were not installed.

Figure 2-20 Sample ASR Logic for DFSMShsm RECALLs and RECOVERs

```
IF &VAMENVIR = 'ACS'
THEN
DO
        IF &HSMFUNC = 'RECALL' OR &HSMFUNC = 'RECOVER'
        THEN
        DO
                IF &STORCLAS = ''
                THEN
                DO
                        SET &STORCLAS     =      'SAMSALSC'
                END
        END
END
```

Technical Overview

BrightStor CA-Allocate is able to choose a new volume (redirect) for allocations done by any of the following DFSMShsm functions:

- BACKUP
- MIGRATE
- RECALL
- RECOVER

BrightStor CA-Allocate recognizes that the allocation is being done by DFSMShsm and modifies DFSMShsm control information as needed to allow DFSMShsm to use the volume that was selected.

Data Set Names Generated by DFSMShsm During MIGRATE and BACKUP

DFSMShsm allocates non-VSAM data sets for MIGRATE and BACKUP with names of the following form:

```
prefix.function.Thhmmss.dsnamefragment
```

prefix -- Installation-defined

function -- **BACK** for BACKUP, **HMIG** for MIGRATE

Thhmmss -- Time of operation

dsnameframent -- A portion of the data set name

DFSMShsm During RECALL/RECOVER

RECALL/RECOVER When DFSMShsm is the Data Mover

When DFSMShsm is the data mover during RECALL or RECOVER, it creates a temporary data set with a name of the following form:

```
prefix.MDB.Thhmmss.dsnamefragment
```

prefix -- Installation-defined

Thhmmss -- Time of operation

dsnameframent -- A portion of the data set name

DFSMShsm then moves the data into this temporary data set. When the data movement is completed, it renames the data set to the real name it is supposed to have. All of these operations are visible to BrightStor CA-Allocate. For example, for a non-VSAM non-SMS data set, BrightStor CA-Allocate typically sees the following ASR environments:

1. ACS for the real name.
2. ALLOC for the temporary name.
3. SPACE for the temporary name.
4. EXTEND for the temporary name during data movement.
5. RENAME during the rename from the temporary name to the real name.

ASR Variables for HSM RECALL and RECOVER Operations

Like all DFSMShsm data set operations, RECALL and RECOVER occur in the DFSMShsm address space, so BrightStor CA-Allocate ASR variables &JOB, &USER, and &GROUP contain information about the DFSMShsm "job", "user", and "group." Information for the requesting job, user, and group are provided in &HSMJOB, &HSMUSER, and &HSMGROUP, respectively.

During a DFSMShsm RECALL or RECOVER when DFSMShsm is the data mover, a temporary data set is allocated, the data is moved, and then the data set is renamed to the real name. Since the original allocation occurs using the temporary name, BrightStor CA-Allocate ASR variables in the ALLOC, SPACE, and EXTEND environments contain information about that name. Special variables are available for DFSMShsm RECALL and RECOVER operations to provide information about the real data set.

The table below lists the relevant variable values during a DFSMShsm RECALL or RECOVER when DFSMShsm is the data mover. See "**Description of ASR Variables**" in the chapter "**Implementation**" for more information about these variables.

TABLE 2-8 ASR Variables when DFSMShsm is the Data Mover

ASR	Variable Value
&DSN	Data set name (temporary name)
&LLQ	Low level qualifier (temporary name)
&HLQ	High level qualifier (temporary name)
&NQUAL	Number of qualifiers (temporary name)
&JOB	"Job" name for HSM address space
&USER	"user" name for HSM address space
&GROUP	"group" name for HSM address space
&HSMDSN	Data set name (real name)
&HSMLLQ	Low level qualifier (real name)
&HSMHLQ	High level qualifier (real name)
&HSMJOB	Job name of requesting job
&HSMUSER	User name for requesting job
&HSMGROUP	Group name for requesting job

During a DFSMShsm RECALL or RECOVER when DFSMSdss is the data mover, there is no temporary name created. All data set operations occur using the real data set name. For consistency, the special-purpose &hsmxxx variable names contain appropriate values that match their corresponding normal variable values. The table below shows these values. See the section **Description of ASR Variables** in the chapter "**Implementation**" for more information about these variables.

Table 2-9 ASR Variables when DFSMSdss is the Data Mover

ASR Variable	Value
&DSN	Data set name (real name)
&LLQ	Low level qualifier (real name)
&HLQ	High level qualifier (real name)
&NQUAL	Number of qualifiers (real name)
&JOB	"job" name for HSM address space
&USER	"user" name for HSM address space
&GROUP	"group" name for HSM address space
&HSMFUNC	HSM function name: 'RECALL' or 'RECOVER'
&HSMDSN	Data set name (real name)
&HSMLLQ	Low level qualifier (real name)
&HSMHLQ	High level qualifier (real name)
&HSMJOB	Job name of requesting job
&HSMUSER	User name for requesting job
&HSMGROUP	Group name for requesting job

The DFSMShsm RECALL Exit (ARCRDEXT)

Installations currently running a DFSMShsm RECALL exit (module ARCRDEXT) need to take some special care:

- A volume selected by BrightStor CA-Allocate overrides any volume selected by the RECALL exit.
- BrightStor CA-Allocate is a better and more complete way to control DFSMShsm allocations because it intercepts all requests and can redirect them to any volume. The DFSMShsm RECALL exit only intercepts non-specific RECALL requests, ignoring those that have a volume specified (directed RECALLs), and the DFSMShsm RECALL exit is restricted to the subset of volumes that DFSMShsm gives it to choose from.

Controlling Only RECALLs and RECOVERs

An installation can have BrightStor CA-Allocate operate only on DFSMShsm allocations done for RECALL and RECOVER operations and leave MIGRATE and BACKUP alone. To stop BrightStor CA-Allocate from controlling BACKUP and MIGRATION allocations, add the following statement to the beginning of your ASR:

```
IF &DSN(2) = 'BACK' OR &DSN(2) = 'HMIG' THEN EXIT CODE(0)
```

Warning: BrightStor CA-Allocate redirects a DFSMShsm migration data set to any volume the ASR decides, and the MIGRATE works. However, if the volume is not defined to DFSMShsm as a migration volume, any subsequent recall fails with the following messages:

```
ARC1010I SYSTEM ACTION AGAINST A MIGRATED DATA SET FAILED
ARC1001I data.set.name RECALL FAILED, RC=0003, REAS=0008
ARC1103I MIGRATION/BACKUP/DUMP VOLUME NOT AVAILABLE
```

If this problem occurs, perform the following steps to define the volume as a migration volume.

1. If the volume is currently defined to DFSMShsm as a PRIMARY or BACKUP volume, issue the DFSMShsm DELVOL command to remove the volume from DFSMShsm.
2. Issue the DFSMShsm ADDVOL command to define the volume to DFSMShsm as a MIGRATION volume. Any recalls against the migration data set now works.

FDR Support

BrightStor CA-Allocate has a feature that allows the user to redirect data set allocations associated with 3 Innovation FDR commands:

1. FDRCOPY — COPY and MOVE
2. FDRDSF — RESTORE

To accomplish this, BrightStor CA-Allocate invokes an FDR documented interface to control which volumes should be used for these allocations. This API passes information about the current data set back and forth between FDR and BrightStor CA-Allocate. This information is used to determine which volume should be used.

Note: Installation instructions are located under the topic "**PLSOPT12**" in the chapter "**Implementation**."

When this support is activated, the following variables are available as INPUT in the ALLOC environment:

&ACCT_JOB	&IFALREADYCAT	&LSPACE_PCTAFT_CYL	&PROCSTEP
&ACCT_STEP	&IFALREADYCATVOL	&LSPACE_PCTAFT_CYLNOT	&PROGRAM
&ALLVOL	&JOB	&LSPACE_PCTAFT_MB	&SECONDARY
&ANYVOL	&JOBCLASS	&LSPACE_PCTAFT_MBNOT	&SPACTYPE
&AVGBLK	&LARGEST_EXTCYL	&LSPACE_PCTAFT_TRK	&STEPNAME
&CONTIG	&LARGEST_EXTCYL_NOT	&LSPACE_PCTAFT_TRKNOT	&SYSID
&DISP	&LARGEST_EXTMB	&LSPACE_RETURN_CODE	&TIME
&DSN	&LARGEST_EXTMB_NOT	&MODULE	&UNIT
&DSORG	&LARGEST_EXTTRK	&NQUAL	&VAMENVIR
&DSTYPE	&LARGEST_EXTTRK_NOT	&NVOL	&VAMRLSE
&GROUP	&LLQ	&PGMRNAME	&XMODE
&HLQ	&LSPACE_FREESPACE	&PRIMARY	

Also, the following variables are available as OUTPUT in the ALLOC environment:

&C0 - &C40	&LSPACE_STORGRP	&LSPACE_VOLUME	&N0 - &N40
&POOLSUB	&STORGRP		

Limitations

BrightStor CA-Allocate is subject to any limitations imposed by FDR. The following is a list of know limitations:

- VSAM and SMS-managed data sets are not supported
- Redirecting a data set to the original source volume is not permitted
- Fragmented data sets that have out of order extents cannot be moved to a smaller device type
- RESTOREing from a 3390 device to a 3380 device is an FDR limitation.

The following sample ASR statements shows how to incorporate this FDR support into your environment.

FIGURE 2-21 Sample ASR that incorporates the FDR Support

```
File   Edit   Confirm   Menu   Utilities   Compilers   Test   Help
------------------------------------------------------------------------------
EDIT                .ASR.PARMLIB(FDR) - 01.02                Columns 00001 00072
Command ==> _____________________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************
000001  IF &VAMENVIR = 'ALLOC' THEN
000002  DO
000003     IF &FDR = 'Y' THEN
000004     DO
000005        SET &UNIT = 'SYSALLDA'
000006        SET &LSPACE_STORGRP = 'SMALL'
000007        IF &LSPACE_RETURN_CODE = 0 THEN
000008        DO
000009           IF &SPACTYPE = 'TRK' AND IF &PRIMARY < &LARGEST_EXTTRK THEN
000010              SET &STORGRP = 'SMALL'
000011           ELSE
000012              IF &SPACTYPE = 'CYL' AND IF &PRIMARY < &LARGEST_EXTCYL THEN
000013                 SET &STORGRP = 'SMALL'
000014              ELSE
000015                 SET &STORGRP = 'LARGE'
000016        END
000017     END
000018  END
```

Space Type Conversions

BrightStor CA-Allocate has a feature that allows the original space allocation unit and quantity to be altered. This can be done for NonVSAM allocations and either at the cluster or component level for VSAM ones. A space type conversion can involve up to as many as ten variables:

All NonVSAM and all VSAM

AVGBLK AVGRECSIZE AVGRECTYPE BLKSIZE OPTBLK SPACCVT

NonVSAM and VSAM Clusters

PRIMARY SECONDARY SPACTYPE UNITTYPE

VSAM Data Components

PRIMARYD SECONDARYD SPACTYPED UNITTYPED

VSAM Index Components

PRIMARYI SECONDARYI SPACTYPEI UNITTYPEI.

The values in the AVGRECTYPE, AVGRECSIZE, AVGBLK, BLKSIZE, OPTBLK, UNITTYPE, UNITTYPED, and UNITTYPEI variables are not changed. Their values, and the value of the SPACTYPE, SPACTYPED, and SPACTYPEI variables, have an impact on how the variables dealing with the primary and secondary allocation quantities (PRIMARY, PRIMARYD, PRIMARYI, SECONDARY, SECONDARYD, and SECONDARYI) are modified.

The remainder of this section contains a general overview of how and why space type conversions are done. Detailed information on each of the above variables as well as examples on how each one participates in a space type conversion is found in **TABLE 4-3 List of Variables** found in the chapter "**Implementation**."

Allocation requests can be made in units of tracks, cylinders, blocks, average records, and in some cases kilobytes or megabytes. Disk devices have different storage capacities. Storage Administrators want to use available disk space efficiently. This is often made easier by converting various allocation units to one standard.

For example, a primary allocation quantity of ten, without knowing whether it is in blocks, tracks, or cylinders, doesn't give the Allocation Selection Routine enough information to make a pooling decision based on the actual number of bytes requested.

The following statements allow the ASR to make pooling decisions based on the actual amount of space being requested.

FIGURE 2-22 Sample ASR that Allows Pooling Decisions

```
File   Edit   Confirm   Menu   Utilities   Compilers   Test   Help
------------------------------------------------------------------------
EDIT           .ASR.PARMLIB(POOLDESC) - 01.01               Columns 00001 00072
Command ==> ____________________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************
000001  SET &UNITTYPE = '3380'
000002  SET &SPACTYPE = 'TRK'
000003  SET &PRIIMARY = &PRIMARY
000004  SET &SECONDARY = &SECONDARY
000005  SET &N0 = (( &PRIMARY ) + (&SECONDARY = 15))
000006  IF &N0 > 100 THEN SET &STORGRP = 'LARGE'
000007  ELSE IF &N0 > 25 THEN SET &STORGRP = 'MEDIUM'
000008       ELSE IF &NO > 5 THEN SET &STORGRP = 'SMALL'
000009            ELSE SET &STORGRP = 'TINY'
```

The above ASR code causes BrightStor CA-Allocate to do a "real time" conversion of whatever the original allocation unit was (that is, BLK, CYL, and so on) to the equivalent amount of 3380 tracks. This is the default action of the SPACCVT variable ('Y': do the space type conversion).

Note: To convert to BLK, a BLKSIZE must be available. If there is no BLKSIZE specified, the conversion is to the appropriate allocation in TRK, which causes space to be released on track boundaries as expected. In this case, TRK allocations are unchanged and CYL is converted to TRK.

In some cases, the Storage Administrator wants to set both the allocation unit and quantity to a fixed amount. In such cases, in which conversion is not required, set the value of the SPACCVT variable to 'N', as in the following ASR code converts all temporary data set allocation requests to SPACE=(TRK,(25,25)).

```
IF &DSTYPE = 'TEMP' THEN
   DO
       SET &SPACCVT = 'N'
       SET &SPACTYPE = 'TRK'
       SET &PRIMARY = 25
       SET &SECONDARY = 25
   END
```

Chapter 3

Installing

This chapter explains installing BrightStor CA-Allocate. This includes a step-by-step procedure to install the product using SMP/E and detailed instructions for tailoring the cataloged procedure.

Getting Started

Before installing any portion of BrightStor CA-Allocate, it is important to become thoroughly familiar with its capabilities. We recommend that you study the chapters "**Concepts and Facilities**s", "**Implementation**", "**Operation**", and if applicable, the chapter "**Quota**". Although this product can be installed with minimal options in effect, it is best to know its total capabilities when planning an implementation.

Hardware Requirements

One 3490 tape drive is required for installation. BrightStor CA-Allocate has no DASD or TAPE unit dependencies, and requires minimal CPU processor capacity. Approximately 30 cylinders of 3390 DASD space are required for installation.

Software Requirements

BrightStor CA-Allocate is supported by the following levels of IBM software:

- OS/390 Version 1 Release 1 or later
- z/OS Version 1 Release 1 and later

During execution, BrightStor CA-Allocate determines what level of software is running and takes the appropriate action.

BrightStor CA-Allocate is written in assembler language, with the exception of the reports, which are written in SAS. SMP/E is the only software product required to install or operate BrightStor CA-Allocate.

Installation

BrightStor CA-Allocate installation with SMP/E is done using the Receive-Apply-Accept method. SMP/E invokes the linkage editor and IEBCOPY to download the distribution tape, and to install it into the target and distribution libraries.

APF Authorization

An APF authorized library for BrightStor CA-Allocate to operate from needs to be established.

SMP/E Process

We have provided sample jobs to set up a complete SMP/E environment. If you already have the SMP/E environment established, you can skip the steps that you do not need to perform.

All of the required SMP/E data sets are internally defined as DDDEF entries. There is no need for an SMP/E proc. All you need to do is either execute the SMP program using "PARM=`CSI=...." as documented in the SMP/E Reference manual, or you can use the ISPF interface provided with SMP/E.

Note: BrightStor CA-Allocate (version 5.1 or later), and BrightStor CA-Vantage Storage Resource Manager (BrightStor CA-Vantage) should be installed into the same target and distribution zones. Do not install into the same zones as BrightStor CA-Disk, other vendor's products, or your operating system, because naming conflicts can occur.

Step 1. Preparing for Concurrent Use of One or More Releases

You can install this release of BrightStor CA-Allocate and continue to use an older release for your production environment. If you do plan to continue to run a previous release, consider the following:

1. Installing this release automatically deletes previous releases. If you are installing this release into a zone that contains an older release, the older release is deleted.
2. For your new release, you must select different target and distribution zones from where your current release is installed.
3. You define DDDEF entries in your new zones to point SMP/E to the proper libraries for installation. Make sure that they point to the new release libraries.

Step 2. Download the Related Installation Materials Library

The JCL needed to install BrightStor CA-Allocate is provided in the Related Installation Materials library (RIMLIB) on the distribution tape (file 2). Important information related to the installation process is provided in the Program Directory library (PGMDIR) on file 4. Run the following JCL to allocate both of these libraries and download them from tape.

FIGURE 3-1 Sample RIMLIB JCL

```
File   Edit   Confirm   Menu   Utilities   Compilers   Test   Help
------------------------------------------------------------------------
EDIT                          (RIMLIB) - 01.00               Columns 00001 00072
Command ==> ____________________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************
000100  //JOBCARD  JOB (####,###),'NNNNNNNN',CLASS=X,MSGCLASS=T,NOTIFY=&USERID
000200  //COPY     EXEC  PGM=IEBCOPY,REGION=4096K
000300  //SYSPRINT  DD   SYSOUT=*
000400  //SYSUT1   DD DSN=RIMLIB,DISP=(OLD,PASS),
000500  //           UNIT=3490,VOL=(,RETAIN,,SER=VVVVVV)
000600  //           LABEL=(2,SL)
000700  //SYSUT2   DD DSN=YOUR.RIMLIB,DISP=(,CATLG,DELETE),
000800  //           UNIT=SYSDA,SPACE=(TRK,(15,3,6),RLSE)
000900  //SYSUT3   DD UNIT=SYSDA,SPACE=(CYL,1)
001000  //SYSUT4   DD UNIT=SYSDA,SPACE=(CYL,1)
001100  //SYSIN  DD DUMMY
001200  /*
001300  //COPY2 EXEC PGM=IEBCOPY,REGION=512K
001400  //SYSPRINT DD  SYSOUT=*
001500  //SYSUT1    DD DSN=PGMDIR,DISP=(OLD,KEEP),LABEL=(4,SL),
001600  //             UNIT=3490,VOL=SER=VVVVVV
001700  //SYSUT2    DD DSN=YOUR.PGMDIR,
001800  //             UNIT=SYSDA,DCB=(RECFM=FBA,BLKSIZE=3120,LRECL=80),
001900  //             SPACE=(3120,(20,5,27),,,ROUND),DISP=(NEW,CATLG)
002000  //SYSUT3    DD   SPACE=(CYL,(2,2)),DISP=(NEW,DELETE),UNIT=SYSDA
002100  //SYSUT4    DD   SPACE=(CYL,(2,2)),DISP=(NEW,DELETE),UNIT=SYSDA
002200  //SYSIN    DD DUMMY
```

Note: SMP/E is equipped with ISPF panels that can assist you in downloading the RIMLIB and PGMDIR libraries.

The RIMLIB data set contains jobs to install the base product with SMP/E. Customization jobs are still located in the installation library.

Full Product Distribution Tapes contain the base release and SMP/E format PTFs. If there are PTFs on the distribution tape, you can receive them at the same time you received the base product. However, you should apply and accept the base release before you apply any PTFs.

Step 3. Prepare the SMP/E Environment

Customize and submit RIMLIB member SMP01GBL to define the global CSI data set, allocate the SMPPTS and SMPLOG data sets, and initialize the global zone.

Customize and submit RIMLIB member SMP02TGT to define the target CSI data set, allocate the SMPMTS, SMPSCDS, and SMPSTS data sets, and initialize the target zone.

Customize and submit RIMLIB member SMP03DLB to define the distribution CSI data set, and initialize the distribution zone.

Step 4. Allocate Data Sets

To allocate BrightStor CA-Allocate data sets, customize and submit RIMLIB data set member ALC01ALC to create target and distribution data sets for BrightStor CA-Allocate.

Step 5. Set DDDEF Entries

Customize and submit RIMLIB data set member ALC02DDD to create DDDEF entries for BrightStor CA-Allocate in the target and distribution zones.

Step 6. Receive BrightStor CA-Allocate

Customize and submit RIMLIB member ALC03REC to receive BrightStor CA-Allocate. If there are PTFs on the distribution tape, they are also received at this time. Do not apply the PTFs until after you have accepted the base function for BrightStor CA-Allocate.

You can use RIMLIB member RECEIVE to receive all sysmods and hold data from the distribution tape.

Step 7. Apply BrightStor CA-Allocate

Customize and submit RIMLIB member ALC04APP to apply the base function for BrightStor CA-Allocate.

Step 8. Accept BrightStor CA-Allocate

Customize and submit RIMLIB member ALC05ACC to accept the base function for BrightStor CA-Allocate.

Step 9. Apply Maintenance

Maintenance to BrightStor CA-Allocate is shipped on the distribution tape and is received at the same time as the Base function. Apply this maintenance now.

Customize and submit RIMLIB member ALC06PTF to apply the PTFs. You can accept the PTFs according to your own installation's policy.

When you apply maintenance, you normally encounter SMP/E hold data. The latest hold data file from the esupport.ca.com web-site should be downloaded and received with whatever maintenance or installation tape is being received.

There are two different types of hold data:

1. **Internal hold data** — Data that is an instream part of the sysmod instructing you of special conditions:

   ACTION You must perform special processing either before or after you apply this sysmod.

   DEP There is a dependency for this sysmod that you must externally verify.

   DELETE This sysmod deletes a load module. You cannot reverse this type of sysmod with the SMP/E RESTORE command.

   DOC There is a documentation change with this sysmod.

   EC This sysmod requires a hardware engineering change. An EC hold sysmod usually does not have an effect on the product unless the EC is present on the hardware device.

   UCLIN You need to perform a UCLIN either before or after you apply this sysmod.

   You must code a bypass operand on your APPLY command to install sysmods that have internal holds. You should only code the bypass operand after you have performed the required action, or if you are performing the action after they apply, if that is appropriate.

2. **External hold data** — External hold data is not a part of the PTF. It resides in a separate file. On SAMS product tapes, there is a HOLDDATA file. External hold data is usually used for sysmods that have been distributed, and later are discovered to cause problems.

   To take advantage of the external hold data, you must receive it into your SMP/E environment. If you use the jobs supplied by Computer Associates, SMP/E receives the hold data.

   If a sysmod has an unresolved hold error, SMP/E does not install it unless you add a bypass to your apply command. You may want to bypass an error hold in situations that do not affect you. This may be a problem that only happens with a hardware device that you do not have, or in a product feature that you do not use.

   When Computer Associates issues the sysmod that resolves the hold, the resolving sysmod supersedes the hold error. This allows you to apply the original sysmod in conjunction with the fixing sysmod.

   There is a special hold data class called ERREL. This means that Computer Associates has determined that the problem fixed by the sysmod is more important than the one that it causes. Computer Associates recommends that you apply these sysmods.

   The easiest and most reliable way to manage external hold data is to allow SMP/E to manage it automatically. When you allow SMP/E to manage the process, the only manual task you need to do is running a REPORT ERRSYSMODS. This report identifies any held sysmods that you may have already applied to your system. If the resolving sysmod is in receive status, SMP/E identifies the sysmod that you need to apply to correct the situation.

   Occasionally maintenance to BrightStor CA-Allocate requires simultaneous maintenance to the BrightStor CA-Vantage Storage Resource Manager component that is distributed with it. The two PTFs will be distributed together via a SMP/E IFREQ. The latter PTF may include updates to the PARMDEF member residing in the SMP/E SAMPARM DDDEF as well as the usual OBJ members making up the LMODs residing in the ACSVLOAD DDEF.

   Since it is unlikely you actually run BrightStor CA-Allocate from the PARMLIBs and LOADLIBs built from SMP/E, you will need to copy updates from the SMP/E libraries into others. Copying the entire contents of all of the SMP/E libraries updated by the PTFs will free you from the trouble of trying to manually determine which members just got updated. This practice will also eliminate the possibility of a partial change being propagated into production libraries.

Step 10. Activate the SUs

Note: Before running the ALLOCSET utility, ensure that the parmlib data set contains member CONFIG. If not, member SAMPCONA should be renamed to CONFIG. There is no need to edit the CONFIG member.

You must activate your Selectable Units (SU) before you can use them. You must run the ALLOCSET job with the APK code(s) you received for the SU for which you have the license. For example,

```
//JOBNAME JOB (ACCT INFO)
//*-------------------------------------------------------------------------*
//* THE PROGRAM MUST BE RUN FOR THE BASE SYSTEM (THE ALLOCATE SUBSYSTEM) *
//*                      AND FOR EACH SELECTABLE UNIT (SU).*
//* SET INIT TO THE FIVE CHARACTER TRIAL OR PERMANENT LEASE CODE WHICH *
//*                      CAN BE REQUESTED FROM YOUR SALES REPRESENTATIVE.*
//* SET ZCMP TO YOUR COMPANY NAME (MAXIMUM 40 CHARACTER).*
//*-------------------------------------------------------------------------*
//UPDATESU EXEC PGM=GENPGM00,PARM=('INIT=XXXXX',
//             'ZCMP=YOUR COMPANY NAME')   <== MUST START IN POS16
//*-------------------------------------------------------------------------*
//* SET THE TRIAL/PERM.LEASE CODE AND CUSTOMER NAME FOR ONE COMPONENT *
//*-------------------------------------------------------------------------*
//STEPLIB   DD DSN=YOUR.SAMS.LOADLIB,DISP=SHR
//PARMS     DD DSN=YOUR.SAMS.PARMLIB,DISP=SHR
```

Expired License

If you are using an evaluation copy of BrightStor CA-Allocate for a limited period of time, at the end of the evaluation period you see the following messages at the operator console:

```
VAM0400 ALL SELECTABLE UNITS HAVE EXPIRED. CA-ALLOCATE CANNOT BE INSTALLED.
VAM0402 HAS AN INVALID EXPIRATION DATE AND CANNOT BE INSTALLED. PLEASE CONTACT
        YOUR MARKETING REPRESENTATIVE FOR ASSISTANCE.
```

Contact your sales representative for instructions on how to activate BrightStor CA-Allocate permanently. Use the sample JCL member ALLOCSET in the Install library to submit an Evaluation or Permanent Activation Code received from your sales representative.

Tailoring the Cataloged Procedure

A cataloged procedure, listed in Figure 3-2, is provided to allow BrightStor CA-Allocate to be implemented as a Started Task. It needs to be modified to meet an installation's requirements. Also ensure that the dsname of the BrightStor CA-Allocateloadlib, is in your PROGxx member of SYS1.PARMLIB. The BrightStor CA-Allocate load library must be APF authorized. To avoid 806 abends, never remove STEPLIB, because the started task does LOADs specifically from STEPLIB.

The Cataloged Procedure is distributed in the installation library as follows:

FIGURE 3-2 The Cataloged Procedure

```
File   Edit   Confirm   Menu   Utilities   Compilers   Test   Help
------------------------------------------------------------------------
EDIT   SYS1.PROCLIB(VAM)  - 01-00                          Columns 00001 00072
Command ==> _____________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************
000001  //VAM PROC
000002  //*********************************************************************
000003  //*                       CA-Allocate                                 *
000004  //*                SMS PLUS FOR CA-VANTAGE                            *
000005  //*                                                                   *
000006  //*  © COPYRIGHT 2001 COMPUTER ASSOCIATES                             *
000007  //*     ALL RIGHTS RESERVED                                           *
000008  //*     LICENDED MATERIAL, PROPERTY OF COMPUTER ASSOCIATES            *
000009  //*                                                                   *
000009  //* ALL RIGHTS RESERVED BY COMPUTER ASSOCIATES. NO PART OF THIS       *
000010  //* PRODUCT MAY BE REPRODUCED, STORED IN A RETRIEVAL SYSTEM OR        *
000011  //* TRANSMITTED IN ANY FORM OR BY ANY MEANS, INCLUDING PHOTOCOPYING,  *
000012  //* RECORDING, ELECTRONIC, MECHANICAL OR OTHERWISE WITHOUT THE        *
000013  //* WRITTEN CONSENT OF THE COPYRIGHT HOLDER.                          *
000002  //*********************************************************************
000014  //VAM       EXEC  PGM=VDSST451,TIME=1440,REGION=6M
000015  //STEPLIB    DD   DISP=SHR,DSN=VAM.RVRM.VDSLOAD
000016  //PARMS      DD   DISP=SHR,DSN=VAM.RVRM.VDSPARM
000017  //SYSPRT     DD   SYSOUT=*
000018  //SYSABEND   DD   SYSOUT=*
```

Notes for Figure 3-2:

- Change the data set name on STEPLIB to a data set used in your shop.
- Change the data set name on PARMS to the data set that contains your System Operational Parameters (that is; VKGPARMS, PARMDEFS, CONFIG, and MESSAGES).

  Note: The PARMS DD statement is also used by BrightStor CA-Vantage. If you are using BrightStor CA-Vantage and BrightStor CA-Allocate, both products should point to the same data set.

The installation process requires the creation of several parameter files, listed below in Table 3-1. This process is described in more detail in the chapter “Implementation.” The INSTALL and PARMLIB data sets contain samples of these members. These members can be placed in any library with LRECL=80, so long as the library(s) are referenced by the appropriate DD statements in the catalog procedure shown in Figure 3-2.

TABLE 3-1 Parameter File Member Descriptions

Member	Location	Description
VKGPARMS	PARMLIB	Operations specifications
PARMDEFS	PARMLIB	Default Operation Specifications
CONFIG	PARMLIB	Required for startup
MESSAGES	PARMLIB	Default Message Specifications
VDSPROG	INSTALL	Data set Operation ASR
VDSTORGP	INSTALL	Storage Group definitions
QREBUILD	INSTALL	Volume selection ASR for the QREBUILD process
QSCAN	INSTALL	Data set Operations ASR for the QSCAN process
QCONFIG	INSTALL	Quota Configuration File

Notes for Table 3-1:

1. The VKGPARMS, MESSAGES, CONFIG, and PARMDEFS members must all be in the PARMLIB pointed to by the PARMS DD of the BrightStor CA-Allocate Started Task. However, only VKGPARMS is intended to be edited by the storage administrator. **The MESSAGES and PARMDEFS are shipped as they are intended to be used.** If you want more information about these members, and you have BrightStor CA-Vantage, refer to the *BrightStor CA-Vantage Storage Resource Manager User Guide*.
2. The QREBUILD, QSCAN, QCONFIG members and the DISKQTBL DD statement are required only if you have the Quota selectable unit.
3. The members in the INSTALL partitioned data set can be given any name. See the chapter "**Implementation**" for information on how to change these names.

Chapter

4

Implementation

This chapter describes how to use BrightStor CA-Allocate to enforce allocation standards and to monitor and optionally enforce allocation limits. Several sets of installation-defined parameters are used to accomplish these objectives.

The parameters are members of partitioned data set(s) as follows:

- Operations specifications definition (member VKGPARMS)
- Storage Group (Pool) definitions (member VDSTORGP)
- Data set Allocation Selection Routine (member VDSPROG)
- Quota Configuration File (member QCONFIG)
- Quota Table Rebuild ASR (member QREBUILD)
- VTOC Scan ASR (member QSCAN)

Creating the Operations Specifications Definition

Most data sets, DD cards, operator commands, and special processing flags are pre-defined in the PARMDEFS member. However, any of these items can be overridden in the VKGPARMS member to customize the BrightStor CA-Allocate environment.

The storage administrator changes default values in the VKGPARMS member. An example showing how to code this member is found in member SAMPVKGP in the library associated with DDDEF SAMSAMP.

To activate the new sysparm values, use PARMREF.

Each parameter is described below, providing the valid values, default values, and overrides (if available).

VKGPARMS Parameters

SUPLS

If you are licensed to run the SMS Plus selectable unit of BrightStor CA-Vantage, specify (Y) to activate this functionality. The default value is (N).

SUPLSNV

If you are licensed to run the Allocation Manager selectable unit of BrightStor CA-Allocate, specify (Y) to activate this functionality. The default value is (N).

SUPLSQ

If you are licensed to run the Quota selectable unit of BrightStor CA-Allocate, specify (Y) to activate this functionality. The default value is (N).

PLS451

This special processing flag should only be changed from the default value when instructed by Computer Associate's support staff. The default value is (Y).

PLS453

This special processing flag should only be changed from the default value when instructed by Computer Associate's support staff. The default value is (Y).

PLSACS

This special processing flag should only be changed from the default value when BrightStor CA-Allocate's ACS Support is to be completely disabled. Specify (Y) to install the VAMACS00 intercept point into the operating system. Specify (N) to prevent the installation of the VAMACS00 intercept point into the operating system. The default value is (Y).

PLSACTN

Specifies the action to be taken by the BrightStor CA-Allocate Started Task, described in the PLSSTCN parameter. The default value is (INSTALL). This action can be overridden by using the Operator commands.

PLSALLVL

Specifies what MVS ALIASLVL to use. Specify (1), (2), (3), or (4). This option is only used in installations making use of the MVS/ESA Multi-Level Alias (MLA) facility. The default value is (1). This can be overridden by using the ALIASLVL= operator command.

PLSARCVL

Indicates the name of the archive volume used by BrightStor CA-Disk. This information is used by BrightStor CA-Allocate during its criteria screening for OLD Environment eligibility. This value must match the value specified for the BrightStor CA-Disk sysparm RECATVOL. The default value is (ARCIVE).

PLSBYPAS

Specifies the DD name used to inactivate BrightStor CA-Allocate in a job step. The default value is (VDSBYPAS).

PLSDIAGD

Specifies the DD name used to produce the special internal diagnostics. The default value is (VDSDIAGS).

PLSDIAGS

Specifies a jobname for which the special internal diagnostics should be issued. This option is useful when it is inconvenient to use the DD card specified with PLSDIAGD, as with DB2 or DFSMShsm. This is the only way to get diagnostics in the ACS environment, when DD cards are unavailable. This can be overridden by the DIAGS=operator command. The default value is ( ).

PLSDQTDS

Specifies the name of the physical sequential data set that contains the Disk Quota Table. The default value is (VAM.DISKQTBL). This data set is only used if the SUPLSQ is (Y) and can be overridden by adding the DISKQTBL DD to the BrightStor CA-Allocate Started Task. If your site is not licensed for the Quota feature, then this parameter has no meaning.

PLSDRMNT

Specifies the DD name used to make BrightStor CA-Allocate process when in dormant state (as described in the PLSMODE parameter). The default value is (#VAMTST#).

PLSENQDS

If PLSOPT1 is set to 'Y', this parameter specifies the data set name used to serialize BrightStor CA-Allocate termination process with BrightStor CA-Disk Auto-Restore jobs.

The value set for this parameter is that of a dsname, the default value being VAM.ENQ.DATA.SET. You should normally choose an unused model data set and you must also reference it in the DMSAR proc as follows:

```
//anydd DD DSN=plsenqds.dsname,DISP=SHR
```

Caution: BrightStor CA-Allocate issues an exclusive SYSDSN ENQ to this data set. If the dsname you choose is a real data set, you must ensure that this enqueue does not cause a resource lock out.

Activating this facility instructs BrightStor CA-Allocate to remove its system hooks in a serialized manner, preventing possible S0C4 abends from occurring within BrightStor CA-Disk Auto Restores. When this facility is activated and BrightStor CA-Allocate is removed from the system, the following occurs:

1. BrightStor CA-Allocate determines whether or not a DMSAR STC is active. This is done immediately before freeing the storage of each of the two SVC26 SVC intercepts.
2. If a DMSAR is active, BrightStor CA-Allocate issues a WTOR to the system console giving the operator the option of waiting for the DMSAR to complete. If the operator chooses to wait, the value for PLSENQDS is enqueued to prevent any new DMSARs from completely initializing.

   Caution: At this point, a resource delay can possibly occur for an incoming DMSAR. The next DMSAR cannot completely initialize because of the SYSDSN ENQ held by BrightStor CA-Allocate. If this situation occurs, the following options are available to you:

   - Reply 'C' to the outstanding WTOR VAM0453 to CANCEL the wait and REMOVE without FREEMAINing.
   - Terminate the WAIT in the END USER ASID that requested the Auto Restore.
   - Cancel the Auto Restore request that started after the REMOVE was issued for BrightStor CA-Allocate.

3. When active DMSARs complete (or if none were active), BrightStor CA-Allocate FREEMAINs the storage of the SVC hooks and continue removing the product.
4. If the option not to wait is chosen, then BrightStor CA-Allocate removes without FREEMAINing the SVC hook. Doing so leaves up to 40K not freed in extended CSA. This process is invoked as needed for each BrightStor CA-Allocate SVC hooks as termination proceeds.

PLSJES3

When redirecting TAPE allocations to DISK, JES3 does not free the tape device that JES3 originally reserved.

JES3 SETUP reserves TAPE devices for every DD statement in the JOB that specifies a tape unit. Specifying this parameter with a value of (Y) ensures that unneeded tape devices are freed at step termination if it is determined that they are not be needed later in the job. The default value is (N).

Note: This is a special processing flag that should only be used in JES3 complexes.

PLSKBYTE

Factor used internally in bytes => Kilobyte => Megabyte => Gigabyte => Terabyte => Petabyte quota calculations. Possible values are ‘1000’ or ‘1024’. Default value is the historical value of ‘1024’. Many software products use ‘1000’ instead of ‘1024’ in similar calculations. This option is being provided for customers where consistency across products is more important than the loss of precision in the quota allocation statistics that will result when ‘1000’ is used.

PLSMODE

Indicates the operation mode of the BrightStor CA-Allocate Started Task. Specify (ACTIVE) to set BrightStor CA-Allocate to the active state, the normal mode of operation. Specify (DORMANT) to set BrightStor CA-Allocate to the dormant state. When in dormant state, BrightStor CA-Allocate processes an allocation only when the DD described in the PLSDRMNT parameter is present. The default value is (ACTIVE). This can be overridden by using either the ACTIVE or DORMANT operator command.

PLSOPT1

Activates the abend prevention facility described in parameter “PLSENQDS.” The default value is (N).

If you specify 'Y', BrightStor CA-Allocate serializes the removal of its SVC hooks with jobs that reference the data set name specified in the PLSENQDS parameter. For detailed information, see the parameter description for PLSENQDS.

Note: This is a special processing flag normally used only in conjunction with BrightStor CA-Disk.

PLSOPT2

Provides support for RACF Discrete profiles during DFSMShsm RECALL and RECOVER operations. The default value is (N). If you specify `Y', BrightStor CA-Allocate's NonVSAM volume selection routine, when it sees that a data set has a RACF Discrete Profile, issues the appropriate RACROUTE requests to prevent 913-38 errors when BrightStor CA-Allocate redirects such allocations.

PLSOPT3

This parameter can be used to change the default action taken during tape to disk conversion with respect to the UNIT=AFF=ddname JCL parameter. The default value is (N).

When (Y) is specified:

- All DD statements that have UNIT=AFF=ddname will have UNIT AFFINITY automatically disabled.
- Space type and space quantities will automatically be propagated from the JFCB associated with the UNIT=AFF=ddname.

PLSOPT4

Reserved for future development. The default value is (N).

PLSOPT5

Reserved for future development. The default value is (N).

PLSOPT6

PLSOPT6 (Y) must be specified in BrightStor CA-Allocate's PARMLIB in order to activate the new EOV Support for IAM file update operations, which is only applicable to releases V6.4/07P and above of IAM. Activating this new support for earlier releases of IAM results in C03 abends when those data sets are closed.

PLSOPT7

Reserved for future development. The default value is (N).

PLSOPT8

Provides the ability to retain the current Quota Water Mark (QWM) values during a QREBUILD operation. The default value is (N). If you specify (Y), BrightStor CA-Allocate's quota table rebuild operations do not reset a quota group's QWM values to the current allocation amount.

PLSOPT9

Defers the actual allocation of new NonVSAM, NonSMS-managed data sets. It is only recommended to use if you have been impacted by the sporadic problem documented as being resolved by PTF SS04993. The default value is (N), not to defer. Specifying (Y) activates the defer logic.

Deferring these allocations increases the window between the time BrightStor CA-Allocate selects a volume because it has sufficient free space and the data set is actually allocated on that volume. This can increase the frequency of "IEF2571 ... - SPACE REQUESTED NOT AVAILABLE" JCL errors because another job has taken the space during this window.

Note: When allocations are deferred via PLSOPT9 (Y), and the ASR exits with a non-zero return code, there will be no special MVS messages issued. It is therefore important, to avoid confusing end users, to have the ASR issue some sort of message indicating that it is choosing to deny this allocation.

PLSOPT10

Allows SMS-managed data set allocations to be intercepted in the ALLOC and DEFINE environments. The default value is (N). Specifying a value of `Y' enables BrightStor CA-Allocate's LSPACE Support to prevent “space not available” failures for new SMS-managed data set allocations.

The way to activate the new sysparm values is using PARMREF.

Support for the following variables, except as noted below, is the same for both SMS and non-SMS-managed data sets:

&BLKSIZE	&DEVCLAS	&DIRBLOCKS	&DSORG	&EXPDT
&LRECL	&NUNIT	&PRIMARY	&RECFM	&SECONDARY
&SPACTYPE	&UNIT	&VLCT		

The following are the differences in support between nonSMS managed and SMS- managed data sets for the above variables:

1. With the exception of LSPACE processing, &UNIT is not an OUTPUT variable in the ALLOC environment for SMS-managed data sets.
2. &DEVCLAS is not an INPUT variable in the ALLOC environment for SMS-managed data sets.
3. &NUNIT is not an OUTPUT variable in the ALLOC environment for SMS-managed data sets. Use &VLCT instead.

Note: Other variables are not supported for SMS-managed data sets in the ALLOC and DEFINE environments.

PLSOPT11

Controls whether or not VSAM support in the EXTEND environment is active. The default value is (N), which deactivates the support. If you specify (Y), it intercepts VSAM extend processing.

When this support is activated, the following variables are available as INPUT with the exception of NVOLD, SECONDARYD and SECONDARYI, which are available as INPUT and OUTPUT:

&ALLVOLD	&ALLVOLI	&ANYVOLD	&ANYVOLI	&C0-&C40
&CISIZE	&CISIZED	&CISIZEI	&DD	&DDD
&DDI	&DSN	&DSND	&DSNI	&EXTENDCOMP
&EXTENTS	&EXTENTSD	&EXTENTSI	&EXTENDVOL	&HLQ
&HLQD	&HLQI	&IMBED	&LLQ	&LLQD
&LLQI	&N0-&N40	&NQUAL	&NQUALD	&NQUALI
&NVOL	&NVOLD"*"	&NVOLI	&PRIMARYD	&PRIMARYI
&RECORG	&REPLICATE	&SECONDARYD"*"	&SECONDARYI"*"	&SPACTYPED
&SPACTYPEI	&UNITD	&UNITI	&UNITTYPED	&UNITTYPEI

"*" These variables are INPUT/OUTPUT.

Tip: Before using these variables look in "Description of ASR Variables" to see any limitations/availability between SMS managed and non-SMS managed data sets.

For details, see the section **VSAM Processing in EXTEND** in the chapter "**Concepts and Facilities**."

PLSOPT12

Controls whether or not the Application Programming Interface for FDR Redirection support is active. The default value is (N), which deactivates the support. If you specify (Y), BrightStor CA-Allocate redirects FDR data set allocations.

After PLSOPT12 has been set to (Y), you must then activate the API within FDR. Choose **one** (1) of the following methods to accomplish this:

- Use the FDR installation dialog on ISPF to set the ALCRSTIN option to YES on panel x.I.4.2
- Execute program FDRZAPOP with the command ZAP ENABLE=ALCRSTIN

Note: The BrightStor CA-Allocate load library must be added to STEPLIB in all FDR jobs. Otherwise you receive an S806 for PLDRSTIN.

Finally, to activate this support, simply include the FDR variable in your ASR. For more information, see the variable description for "FDR."

For related information, see the section **FDR Support** in the chapter "**Concepts and Facilities**."

PLSOPT13

Controls whether or not the Multiple DSCB Open Support for ISPF Allocations is activated. Without this support, ISPF allocations that have more than one open DSCB are bypassed in the EOV environment. The default value of (N) prevents BrightStor CA-Allocate from using this support.

PLSOPT14

Prevents BrightStor CA-Allocate from notifying BrightStor CA-Vantage of threshold violations. The default value of (N) allows BrightStor CA-Allocate to notify BrightStor CA-Vantage of all violations when they occur. Specifying a value of (Y) causes BrightStor CA-Allocate to bypass the notification process.

PLSOPT15

(Y) will install the HSM recall/recover support.

PLSOPT16

Logically enables NonVSAM EOV support for BrightStor CA-Disk OS/390 RESTOREs. The default value is (N).

PLSOPT17

Activates security on the special bypass DDName described in parameter PLSBYPAS. The default value is (N). If (Y) is specified, BrightStor CA-Allocate will query the security system when finding the DDName in a job step, checking the current UserID for authorization to use that DDName. If the UserID does not have authorization, a VAM0038 message will be issued and the job step will be processed by BrightStor CA-Allocate just as if the bypass DDName was not there.

Identify the bypass DDName to your security system prior to activating this support with the following process:

1. Define the specified or default value in the BrightStor CA-Allocate PLSBYPAS system parameter as a generic data set class to the security system with a universal access of NONE. For example, if the default value is used, the generic data set class will be ‘VDSBYPAS.**’
2. CA-Allocate will simulate the allocation of a data set with the DSN comprised of the value of PLSBYPAS and a second node of X. For example, if PLSBYPAS is at the default, the DSN would be 'VDSBYPAS.X.' Refer to your security package documentation for granting the necessary access.

Note: The bypass DDName may be any valid DDName, 1-8 characters in length. It should not be defined as any element (such as an ALIAS) in the master or any other user catalog.

PLSOPT18

Reserved for future development. The default value is (N).

PLSOPT19

Reserved for future development. The default value is (N).

PLSOPT20

This parameter activates the volume selection process to pick a volume randomly. The default value is (N) – not active. Volume selection is done using IBM’s SYSEVENTS DEVALLOC Service.

PLSOPT22

This parameter will cause second choice volume(s) to be chosen from volumes that have the lowest threshold setting in the VDSTORGP parmlib member. The default setting of PLSOPT22 (N) causes the second choice volume(s) to be chosen as described in the section **Volume Selection Processing Logic** in the chapter "**Concepts and Facilities**."

PLSOPT23- PLSOPT93

Reserved for future development. The default value is (N).

PLSOPT94

The default value of N will cause BrightStor CA-Allocate to leave the hooks at IGGPRE00 and IGGPOST0 during a REMOVE operation. This process is referred to as the STUB process. The STUBs provide anchor points for BrightStor CA-Allocate and are effectively dormant when the system is not active. A value of Y would cause the system to remove the hooks in the order they were set. This process is referred to as LIFO (last-in, first-out). Installations running other OEM products that also hook at IGGPRE00 and IGGPOST0 should ensure that PLSOPT94 is set to N to eliminate the LIFO requirement for removal of the system at these locations.

PLSOPT95

The default value is (N). Setting PLSOPT95 (Y) allows the Stop-Not-Cataloged-2 feature to be implemented for non-SMS managed TAPE data sets. See the description of the STOP_NOT_CATLG2 variables later in this chapter. The STOP-NOT-CATLG2 variable can be set to U, R, or D, but if the existing data set is on tape, the only valid option is U. If it is R or D, the SNC2RC variable will be set to 20.

When running with PLSOPT95 (Y), the SPACE environment for TAPE data sets is entered for the sole purpose of allowing an existing tape data set to be uncataloged. Therefore, any of the variables that normally are referenced in the SPACE environment for DASD data sets may not be applicable for TAPE data sets, and should not be referenced. Additionally, any of the variables that can be set in the SPACE environment for DASD data sets cannot be set for TAPE data sets in the SPACE environment. The single exception to this is the STOP_NOT_CATLG2 (SNC2) variable, which can be set as indicated above.

PLSOPT97

Running with the default of (N), creates an SDUMP and other required diagnostic messages when an unusual condition is hit during End-of-Volume processing for VSAM data sets. If an SDUMP occurs, prepare to send the DUMP, complete Allocate STC JES listing and complete job listing where the V37xxxx messages appear to Computer Associates Technical Support. Once you have opened an issue you can change this parameter to "Y" to suppress the dump.

Setting PLSOPT97 to (Y) will suppress the SDUMPs and unconditional messages. Do not suppress the generation of such documentation until at least one set has been captured and sent to Computer Associates Technical Support. Without such "basic DOCs" it will be virtually impossible to diagnose the cause of such problems which may result in abends in other vendors' software later on in the job, long after BrightStor CA-Allocate has completed its EOV Recovery Process.

PLSOPT98

Historically the tracing of jobs via the VDSDIAGS (or other) diagnostic DD statements has resulted in the messages being issued to the SYSLOG as well as the JESMSGLG and JESYSMSG DDs allocated to the applicable batch job or started task. When PLSOPT98 (Y) is specified in PARMLIB, these messages will only be issued to the JESYSMSG DD. The default value of this parameter, (N), will result in these diagnostics being issued in "triplicate" as they historically have been issued.

PLSOPT99

When PLSOPT99 (Y) is specified in PARMLIB, and DIAGS= tracing has been activated for a job or STC, extensive optional diagnostics will be generated. The default value is (N).

PLSPRGDS

Contains the name of the partitioned data set that contains the Allocation Selection Routines, storage pool definitions, and the Quota configuration file members. The default value is (VAM.INSTALL). This data set cannot be overridden by adding the VDSPROG DD to the BrightStor CA-Allocate Started Task because this data set must be dynamically allocated. If you do, an error message is issued telling you to remove it.

PLSQCFMB

Contains the member name that contains the Quota Configuration File that resides in the partitioned data set in the PLSPRGDS parameter. The default value is (QCONFIG). This member name can be overridden by using the QCONFIG= operator command. If your site is not licensed for the Quota feature, then this parameter should be ignored.

PLSQFACT

Units the quota allocation amounts are kept in. Possible values are 'B' (bytes) or 'K' (kilobytes). Default value will be 'B' which results in the historical limitation or 9 terabytes as the maximum quota limit. Customer needing the 9 terabyte limit increased to 9 petabyte should use 'K'. For more information on the use of PLSQFACT, see the section **QUOTA and PLSQFACT** in the chapter "**Quota**" of this guide.

Note: PLSOPT7 no longer provides this function. Use the PLSQFACT (K) function.

PLSQRBMB

Contains the member name that contains the QREBUILD Allocation Selection Routine that resides in the partitioned data set in the PLSPRGDS parameter. The default value is (QREBUILD). This member name can be overridden in the Quota Configuration File (pointed to by the PLSQCFMB member) by using the QREBUILD OPTION. If your site is not licensed for the Quota feature, then this parameter should be ignored.

PLSQSCMB

Contains the member name that contains the QSCAN Allocation Selection Routine that resides in the partitioned data set in the PLSPRGDS parameter. The default value is (QSCAN). This member name can be overridden in the Quota Configuration File (pointed to by the PLSQCFMB member) by using the QSCAN OPTION. If your site is not licensed for the Quota feature, then this parameter should be ignored.

PLSRES

Indicates the residency mode of the BrightStor CA-Allocate Started Task. Specify (Y) to keep the Started Task running waiting for operator commands or specify (N) to terminate the Started Task after the action is completed. The default value is (Y). This can be overridden by using the RESIDENT= operator command.

PLSSABDS

Contains the name of the physical sequential data set where the SYSABEND data should be written. To route this data to an output class, instead of a data set, specify SYSOUT=x, where x is a SYSOUT class. The default value is (VAM.SYSABEND), and sample DCB attributes can be found in member PLSSABDS in the installation library.

This data set can be overridden by adding SYSABEND, SYSUDUMP, or SYSMDUMP DD to the BrightStor CA-Allocate Started Task.

PLSSC

Contains the name of the SMS Storage Class used internally by BrightStor CA-Allocate during the processing of a DFSMShsm RECALL or RECOVER which is not SMS managed. Defining this Storage Class to your SMS configuration is not required. The default value is 'SAMSALSC.' Override the default value only if already using the default in the installation for another purpose.

PLSSG

Contains the name of the SMS Storage Group used internally by BrightStor CA-Allocate during the processing of a DFSMShsm RECALL or RECOVER which is not SMS managed. The default value is 'SAMSALSG.' Override the default value only if already using the default in the installation for another purpose.

PLSSMSEV

Special processing flag that should only be changed from the default value when BrightStor CA-Allocate's End-of-Volume Support is to be disabled for SMS-managed data sets. Specify (Y) to process SMS-managed data sets in the EOV and EOV_VSAM environments. Specify (N) to prevent the processing of SMS-managed data set in the EOV and EOV_VSAM environments. The default value is (Y).

PLSSPRDS

Contains the name of the physical sequential data set where the SYSPRINT (DD name SYSPRT) output should be written. To route this data to an output class, instead of a data set, specify SYSOUT=x, where x is a SYSOUT class.

The default value is (VAM.SYSPRT), and sample DCB attributes can be found in member PLSSPRDS in the installation library. This data set can be overridden by adding the SYSPRT DD to the BrightStor CA-Allocate Started Task.

PLSSTCN

Contains the name of the BrightStor CA-Allocate Started Task. The default value is (VAM). This parameter is required if you wish to submit commands to BrightStor CA-Allocate from within the BrightStor CA-Vantage GUI. If your site is not licensed for BrightStor CA-Vantage Storage Resource Manager, then this parameter can be ignored.

PLSSTRMB

Contains the member name that contains the Storage Group Definition Table that resides in the partitioned data set in the PLSPRGDS parameter. The default value is (VDSTORGP). This member name can be overridden by using the STORGP= operator command.

PLSV37

Special processing flag that should only be changed from the default value when BrightStor CA-Allocate's End-of-Volume Support is to be completely disabled. Specify (Y) to install the EOV intercept points into the operating system.

Specify (N) to prevent the installation of the EOV intercept points into the operating system. The default value is (Y).

PLSV37BY

Indicates the DD name that makes BrightStor CA-Allocate perform special processing during EOV (NonVSAM) environment. The default value is (V37BYPAS).

PLSVDSMB

Contains the member name of the VDSPROG Allocation Selection Routine that resides in the partitioned data set in the PLSPRGDS parameter. The default value is (VDSPROG). This member name can be overridden by using the PROG= Operator Command.

PLSXMEM

Communicates with the BrightStor CA-Vantage subsystem. The default value of (N) shuts off communication with the BrightStor CA-Vantage sub-system. Specify (Y) to retrieve parameters from BrightStor CA-Vantage via a cross-memory service.

PLSXMEM should be set to (N) if you want to use different partitioned data sets for your BrightStor CA-Vantage and BrightStor CA-Allocate parameters. PLSXMEM should be (Y) if you want to use the same partitioned data set and VKGPARMS member for both BrightStor CA-Allocate and BrightStor CA-Vantage. If your site is not licensed for BrightStor CA-Vantage, then this parameter should be ignored.

PLSXRFDS

Contains the name of the partitioned data set that contains the Cross Reference data (XRF), the Cross Reference of each COPYBOOK statement for each ASR. To route this data to an output class, instead of a data set, specify SYSOUT=x, where x is a SYSOUT class, instead of the name of a PDS. The default value is (VAM.XRF), and sample DCB attributes can be found in member PLSXRFDS in the installation library. This data set cannot be overridden by adding the XRF DD to the BrightStor CA-Allocate Started Task because this data set must be dynamically allocated. If you do, an error message is issued telling you to remove it.

PLSZSEC

Activates the additional EOV Support detailed in the **EOV Support for Zero Secondary** section in the "Concepts and Facilities" chapter of this guide. Possible values are 'Y' (activate) or 'N' (disable). The default value is 'N.'

Creating a Storage Group Definition

A storage group is a named set of disk volumes, that is, a "pool" of volumes. A storage group is like an MVS esoteric unit, such as SYSDA. However, one significant difference between a storage group and an MVS esoteric unit is that the storage group definitions can be changed without doing an MVS system generation and an IPL.

Another distinction is that storage groups are formed on the basis of volume serial numbers, whereas MVS esoteric units are based on unit addresses.

Grouping disk volumes simplifies the task of managing allocation requests and disk storage. Instead of managing hundreds of volumes, the Storage Administrator can oversee a handful of storage groups. Every volume to be included in a pool (storage group) must be defined within the storage group definition member. Volumes not defined to any pool are not managed by BrightStor CA-Allocate, so any volume to be managed by it must be defined to some pool. A volume may be defined to more than one pool.

Storage groups are defined by creating a member in a library. The default name for the member is VDSTORGP.

Each 80-byte record contains four column-dependent parts: volume serial number, free-space threshold percentage, include/exclude flag, and storage group names list.

Volume Serial

In columns 1 through 6, enter the volume serial number, left justified, filled with blanks to the right if it is shorter than six characters. The volume serial may contain an asterisk (*), percent sign (%), or a question mark (?) as pattern matching characters. The question mark can only match one numeric digit in the range 0→ 9. The asterisk can match any number of characters from zero to six. The percent sign can only match one character.

Table 4-1 is a sample of how pattern matching works (where x is any valid character):

TABLE 4-1 Pattern-Matching Sample

Pattern	VOLSERs Matched
ISP*	ISP, ISPx, ISPxx, and ISPxxx
LABS5%	LABS5x
*RES	RES, xRES, xxRES, and xxxRES
VOLSE?	Matches if volser is VOLSE0, VOLSE1, … VOLSE9

Free-Space Threshold

In columns 10 and 11, enter the two-digit free-space threshold percentage. The percentage number can be from 00 to 99. The percentage is the target free-space threshold that BrightStor CA-Allocate tries to maintain on the volume. See the section **Volume Selection Processing Logic** in the chapter "**Concepts and Facilities**" for more information on this process.

BrightStor CA-Allocate always chooses to allocate to a volume on which the free-space threshold can be maintained before it chooses one that would violate the target free-space threshold. Some level of free space should be maintained on volumes that have data sets that need to grow into secondary extents. The presence of such a free-space threshold, maintained during primary space allocations, increases the likelihood that space is available when the data sets extend into their secondaries.

If either column contain a blank, a zero is used in its place.

Include/Exclude Flag

In column 15 enter a blank or 'I' to include a volume (or pattern) in a STORGRP. In column 15 enter an 'E' to exclude a volume (or pattern) from a STORGRP. If a volume (or pattern) is both included and excluded from a STORGRP, the exclude overrides the include.

Storage Group Names

In columns 21 through 72, enter a list of one or more storage group names to which the volume or volume pattern belongs. Each name must be separated from the next by a comma (no blanks). If you need more room to add names, simply start another entry with the same volume serial number or pattern.

A storage group name can be from 1 to 8 characters in length. Storage group names are recognized by some other Computer Associates products. BrightStor CA-Disk can recognize them through the VAMPOOL parameter, and BrightStor CA-Vantage recognizes them through the INCLVAMS parameter. Storage group names have no connection with esoteric or generic unit names, volume serial numbers, or anything else outside of BrightStor CA-Allocate. Storage groups can be named with esoteric or generic unit names, or with volume serial numbers, as in the sample in Figure 4-1. A storage group name can be used in the ASR by setting the variable STORGRP using the SET statement, described in "The SET Statement." After a storage group is set in the ASR, volumes are only selected from that storage group.

Syntax for Storage Group Definitions

A blank space in column 1 turns the whole record into a comment. A blank space after a storage group list changes the rest of the record into a comment.

Note: An asterisk in column 1 is considered a leading pattern-matching character of the volume serial field.

Sample Storage Group Definition

Figure 4-1 below shows the sample storage group definition contained in installation library member VDSTORGP.

FIGURE 4-1 Sample Storage Group Definition

```
File   Edit   Confirm   Menu   Utilities   Compilers   Test   Help
------------------------------------------------------------------------
EDIT            .R53.INSTALL(VDSTORGRP) - 01.00            Columns 00001 00072
Command ==> ____________________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************
=COLS>  ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000100   VOLUME   FREE SPACE STORAGE
000200   SERIAL   THRESHOLD  GROUP(S)
000300  ISP*      10         ISPDA      COMMENTS MAY BE PUT HERE
000310  ISP801         E     ISPDA      EXCLUDE ISP801
000400  SCR*      20         SCRATCH
000500  SCR500         E     SCRATCH    EXCLUDE SCR500
000600  LABS5%    20         PRIMARY,3350
000700  LABS5%    20         PRIMARY,3380
000800  *RES      00         SYSTEM
000900  PRF*      00         CACHEDB
001000  WRK00%    30         SCRATCH,MULTI
001100  WRK800    00         SCRATCH,OVERFLOW,3380
001200  MVS*      00         OVERFLOW
001300  CAT*      00         SYSTEM
001400  DBS*      20         DATABASE
001500  DMS%%%    20    I    TEST
001600  DMS9*          E     TEST       EXCLUDE DMS9**
****************************** Bottom  of  Data ********************************
```

In Figure 4-1, eleven storage groups that can later be used (SET) in the ASR are defined. When defining a storage group, a volume serial number (or pattern) is related to a storage group name or names.

Creating an Allocation Selection Routine

An Allocation Selection Routine (ASR) allows Storage Administrators to write simple logic statements that define and enforce their installation's standards for data set allocations and disk space quota limits. The ASR employs a CLIST-like language that gives great flexibility and simplicity in defining how BrightStor CA-Allocate operates. This CLIST-like language allows a simple “program” with IF-THEN-ELSE logic statements to be created. The approach makes obsolete the old-style macros and fixed-field control statement method of specifying options to a software package.

Across all 14 environments, more than 250 variables can be checked. Many of these variables can be modified. For example, during data set allocations, the ASR may change the block size, primary allocation quantity, secondary allocation quantity, and expiration date. In addition, 82 user variables are provided for use as work areas or arguments to a user exit, or to remember the original values of variables you are changing in case you want to cancel the change.

The ASR supports the following statements:

```
FILTLIST name INCLUDE(list) [EXCLUDE(list)]
IF expression THEN statement [ELSE statement]
DO END
SET &variable = expression
WRITE 'text &variable'
EXIT [CODE(number)]
COPYBOOK 'member'
CALL VDSEXIT?
CALL QUOTACHK
```

Each ASR statement is described in detail in the ASR STATEMENTS section.

Syntax for ASR Statements

ASR statements are made up of different parts or components.

The first component is always a keyword, such as FILTLIST, IF, SET, and so on. The keyword tells BrightStor CA-Allocate what this statement does.

The remaining components of an ASR statement are constants, operators, functions, and variables. Each of the components is described in detail in the section on ASR STATEMENTS.

The general syntax for all ASR statements is as follows:

- Each statement must begin with a keyword, and the keyword must be entered entirely in upper case characters.
- Statements can be entered in free format in columns 1 through 72.
- Wherever one blank is allowed, any number of blanks can be used.
- Column 72 of a given line is logically connected to column 1 of the next line.
- Comments are started with a /* and ended with an */. Comments can span multiple lines before reaching the ending delimiter (*/).

  Warning: If a comment (/*) is started but not ended, ALL characters on ALL lines through to the next */ or the end of all statements are considered a single long comment.
- Lower case characters are unequal to their upper case counterparts. “a” and “A” are not the same.

  Warning: All names (variables, keywords, functions) must be entered in upper case, or they are not recognized. Filter list names (FILTLIST), and string (character) constants can be either upper case, lower case, or mixed, but they must be consistent throughout, or they are considered different variables and errors occur.

ASR Statements

The COPYBOOK Statement

```
COPYBOOK 'name'
```

The COPYBOOK statement allows identical blocks of ASR statements to be referenced in multiple places in your ASRs. Use the COPYBOOK statement when the same logic is needed in multiple environments, or to increase the readability of your ASRs.

A COPYBOOK is a member in the partitioned data set referenced by the PLSPRGDS statement in VKGPARMS. The COPYBOOK can contain other COPYBOOKs, up to 20 levels. Recursive COPYBOOKs are not allowed. For example, COPYBOOK 'A' cannot use COPYBOOK 'B' if COPYBOOK 'B' uses COPYBOOK 'A'.

The following examples show one way COPYBOOKs can be used:

- Figure 4-2 contains VDSPROG ASR statements
- Figure 4-3 contains COPYBOOK ‘POOLING’ statements
- Figure 4-4 contains COPYBOOK ‘FILTERS’ statements

FIGURE 4-2 Sample COPYBOOK ASR Statement: ASR.PARMLIB VDSPROG

```
COPYBOOK ‘FILTERS’
IF &USER NE &USERS THEN EXIT CODE (0)
IF &VAMENVIR = ‘ALLOC’ THEN DO
  COPYBOOK ‘POOLING’
  IF &DSTYPE = ‘PERM’ THEN SET &RLSE = ‘Y’
                           END
  IF &VAMENVIR = ‘DEFINE’ THEN DO
    COPYBOOK ‘POOLING’
    SET &INDEXSEP = ‘Y’
                                 END
```

FIGURE 4-3 Sample COPYBOOK ASR Statement: POOLING

```
File   Edit   Confirm   Menu   Utilities   Compilers   Test   Help
-------------------------------------------------------------------------
EDIT              .ASR.PARMLIB POOLING - 01.03            Columns 00001 00072
Command ==> ____________________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************
000001  IF &ANYVOL = &VOLS THEN
000002     DO
000003        SET &FINDVS = 'Y'
000004        SET &POOLSUB = 'Y'
000005        SET &UNIT = 'SYSALLDA'
000006     END
000007   ELSE
000008     DO
000009        SET &STORGRP = 'SSLDA'
000010        SET &FINVS = 'Y'
000011        SET &UNIT = 'SYSALLDA'
000012     END
***************************** Bottom  of  Data ********************************
```

COPYBOOK 'POOLING' Statements

FIGURE 4-4 Sample COPYBOOK ASR Statement: FILTERS

```
File   Edit   Confirm   Menu   Utilities   Compilers   Test   Help
-------------------------------------------------------------------------
EDIT              .ASR.PARMLIB FILTERS - 01.01            Columns 00001 00072
Command ==> ____________________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************
000001  FILTLIST   &USERS   INCLUDE(ISP*)
000002  FILTLIST   &VOLS    INCLUDE(SSL*,DMS9*)
000003                      EXCLUDE(DMS903)
000004  FILTLIST   &PROD    INCLUDE(SYS1.**,SYS2.**)
000005  FILTLIST   &QVOLS   INCLUDE(SSL*)
***************************** Bottom  of  Data ********************************
```

COPYBOOK FILTERS Statements

Note: Do not use a FILTLIST statement if the COPYBOOK is inside a DO-END group.

The FILTLIST Statement

```
FILTLIST name INCLUDE(list) [EXCLUDE(list)]
```

The FILTLIST statement allows the creation of a list of volumes, users, groups, data set names, and so on to be used in a comparison expression. Whenever multiple comparisons are being done against the same variable (like checking USERID against all system programmer names), a FILTLIST statement can simplify and clarify the comparison, making it easier to maintain. The name given to the filter list must start with a letter and have no more than 31 alphanumeric characters. Do not use names that conflict with system unit names or ASR variable names.

The FILTLIST INCLUDE keyword includes items contained in its list, while EXCLUDE excludes items contained in its list. The lists are composed of constants (strings or patterns only) separated by commas. Null values cannot be used. If the variable matches items in both INCLUDE and EXCLUDE lists, it is excluded.

For example, consider the following statement:

```
FILTLIST &WORK INCLUDE(WRK*,SCR*) EXCLUDE('WRK004',SCR00%)
IF &ALLVOL=&WORK THEN EXIT CODE(0)
```

In the preceding example, the ASR is exited when all of the volumes start with WRK or SCR, but continues when it encounters volume WRK004 or any volumes beginning with SCR00.

The following example shows how to continue a filter to multiple lines:

```
FILTLIST &NEWV   INCLUDE (WRK*, MEM*)
                 EXCLUDE (WRK80*,'MEM001','MEM002','MEM003', 'MEM999')
IF &ALLVOL=&NEWV THEN EXIT CODE(0)
```

The ASR is exited when all the volumes specified on the allocation request begin with WRK or MEM, except when it encounters a volume beginning with WRK80, or the volumes MEM001, MEM002, MEM003, or MEM999.

The ampersand (&) before the filter list name is optional when defining the filter list, but is required when using it in a comparison expression. Notice in the above examples that the patterns do not have single quotes around them, and string constants do. See "ASR Constants" for more information on patterns and constants.

Note: The FILTLIST statement is not executable; it merely declares a list. It must appear in the ASR before its first use in an IF statement.

Note: Do not use a FILTLIST statement within any IF construct or DO-END group. This means you cannot code a COPYBOOK inside a DO-END group if that COPYBOOK includes a FILTLIST statement. In general, to avoid problems, place FILTLIST statements first in the ASR.

The IF Statement

```
IF expression THEN statement [ELSE statement]
```

The IF statement first evaluates the expression. If the expression is true (evaluates to a non-zero number), the statement following the THEN keyword is executed. If the expression is false (evaluates to zero), the statement is skipped and an optional ELSE statement is executed.

The statement that is placed after the THEN and ELSE keywords can be simple or compound. A simple statement is one executable statement (such as IF, SET, WRITE, EXIT, or CALL). A compound statement is one or more statements surrounded by a DO-END combination, see the description of The DO Statement below.

The Boolean operators (AND, OR), and comparison operators (equal, not equal, and so on) that can be used in an expression are described later in this section.

The IF statement is the only flow of control statement that is supported. There is no GOTO or WHILE construct, so the flow always runs from the top to the bottom one time during each invocation of the ASR.

The DO Statement

```
DO END
```

The DO statement begins a compound statement. The compound statement ends with the END keyword. Use this construct to do more than one action in an IF statement. Without the DO-END, only one action can take place (a simple statement) in an IF statement. You can actually use DO-END all the time, since one statement within the DO-END delimiters is just as valid as hundreds of statements.

An example of the DO-END is shown below:

FIGURE 4-5 Sample DO-END ASR Statements

```
File   Edit   Confirm   Menu   Utilities   Compilers   Test   Help
------------------------------------------------------------------------
EDIT              .ASR.PARMLIB DO-END - 01.03            Columns 00001 00072
Command ==> ____________________________________________________ Scroll ==> CSR

****** *************************** Top of  Data ********************************
000001  IF &DSN = SYS1.** THEN
000002     DO
000003        WRITE 'ALLOCATING SYSTEM DATA SET &DSN'
000004        SET &STORGRP = 'SYSTEM'
000005     END
000006    ELSE
000007     DO
000008        WRITE 'ALLOCATING NON-SYSTEM DATA SET &DSN'
000009        SET &STORGRP = 'PRIMARY'
000010     END
****** ********************** Bottom  of  Data ********************************
```

The SET Statement

```
SET &variable = expression
```

The SET statement assigns a new value to a variable. Only output variables can be set. The output variables are those designated with an 'O' in one or more environments in TABLE 4-3 List of Variables in the chapter "Implementation." A numeric variable can be set only to a numeric expression. A character variable can be set only to a character expression.

The following are valid SET statements:

```
SET &N0             = 99365         /* numeric        */
SET &C0             = 'SYSALLDA'    /* character      */
SET &EXPDT          = &N0           /* numeric        */
SET &UNIT           = &C0           /* character      */
```

N0 and C0 are user “work” variables, described later in this section.

The output variable STORGRP is special, because it can be set to a “list” of storage group names. Multiple storage group names can be assigned to the STORGRP variable by separating each name with a comma. For example:

```
SET &STORGRP = 'PRIMARY','SCRATCH'
```

The WRITE Statement

```
WRITE 'text &variable'
```

The WRITE statement issues a message to the job or userid requesting the allocation. This enables the ASR to warn users about violations or to tell them about redirections or other changes to their allocations. TSO users receive the message at the terminal. For other jobs and started tasks, the message appears in the job log. In addition, it is possible to direct a message to specific TSO users logged onto TSO when the message is sent. User-specific messages are not saved.

The message to be displayed must be enclosed within single quotation marks ('). It can contain text and / or variables. Text is displayed as written. A variable must begin with a single ampersand. Any variable found in the WRITE statement is replaced with the value contained in the variable. Only as many character positions as are necessary to display the value are taken up on the message line. The total number of characters in the output message line cannot exceed 79 characters. Anything after the first 79 characters is lost.

The statement:

```
WRITE 'DSN = &DSN, DISP = &DISP'
```

may expand to:

```
DSN = TIM1.VERY.VERY.VERY.VERY.LONG.DATASET.NAME, DISP = NEW
```

The maximum length of each character variable is listed in the variable description. For numeric variables, the maximum length is always 10 characters (although for variables like TODAY, which contains a date (89198), the length is always 5 characters).

To display an ampersand (&), or a single quotation mark (') in the output message line, put two ampersands (&&), or two single quotation marks ('') in the WRITE statement text.

The statement:

```
WRITE '&&DSN = ''&DSN'''
```

can expand to :

```
&DSN = 'TIM1.DATASET.NAME'
```

Warning: Subscript operators and the SUBSTR function (described later in this section) are not allowed in the WRITE string. They can, however, be referenced as follows:

```
SET &C0 = &DSN(3)  /*SUBSCRIPTED DATA SET NAME*/
    WRITE 'THIS IS A &C0 TYPE DATA SET'
SET &C1 = &SUBSTR(3:3,&USER)  /*SUBSTR FUNCTION*/
    WRITE 'YOU ARE NOT ALLOWED ON THAT PACK, &C1'
```

The format for sending a message to a specific TSO USERID is:

```
WRITE '@@&USER message text...'
WRITE '@@&HSMUSER message text...'
WRITE '@@userid message text...'
```

Where &USER or &HSMUSER is the USERID referenced by these &variables and “userid” is a specific TSO USERID.

The format for sending a message to a BrightStor CA-Vantage is:

```
write '@#@userid message text...'
```

Where “userid” is the TSO USERID of a BrightStor CA-Vantage user.

The EXIT Statement

```
EXIT [CODE(number)]
```

The EXIT statement causes the ASR to stop executing immediately. When BrightStor CA-Allocate exits the ASR, it writes changed variables back to MVS and returns to the calling environment with any return code that was specified (the default return code is 0). Following the last statement in the ASR, an implied EXIT CODE(0) is executed automatically. If the CODE keyword is omitted, a zero return code is assumed. Only specify code values are given in Table 4-2 .

TABLE 4-2 ASR Return Codes and Their Meaning

Environment	RC	Description
ACS	0	Change SMS constructs.
	other	Do not change SMS constructs, return code &ACSRC, reason code &ACSRSN.
ALLOC	0	Continue with allocation.
	4	Cancel the allocation.
DEFINE	0	Continue with the VSAM define.

Environment	RC	Description
	other	Cancel the request. Return the specified return code as the error return code to the issuer of the Catalog Management request.
EOV EOV_VSAM	0	Attempt to select another volume from the storage group that has either been implied (&POOLSUB) or explicitly specified (&STORGRP). If volume selection is successful, the original out of space condition is avoided.
	4	Allow the job to fail for the original out of space condition.
EXTEND	0	Allow extend.
	4	Abend E37-0C.
OLD	0	Make variable changes.
	other	Ignore all other variable change requests.
QREBUILD	0	Add this volume to the list of volumes scanned.
	4	Do not add this volume to the list of volumes to be scanned.
QSCAN	any	The return code is not used for QSCAN.
QUOTA	0	Continue with request.
	4	This volume serial is rejected.
	8	Cancel the request. Only available if Optional Maintenance PI **601599** is installed.
RELEASE		Same as SPACE.
RENAME		Same as SPACE.
SCRATCH		Same as SPACE.
SPACE	0	Continue with request.
	4	This volume is rejected.
	8	Cancel the request. Only available if Optional Maintenance PI **601599** is installed.

Unsupported values are handled as follows:

- 1, 2, and 3 result in a 0.
- 5, 6, and 7 result in a 4.
- All other numbers result in the highest valid return code. For example 9 becomes 8 in SPACE and 4 in ALLOC.

For the SPACE, EXTEND, RELEASE, SCRATCH, and RENAME environments, the meaning of the return code is defined by IBM for the IGGPRE00 DADSM pre-allocation exit. This is the hook point where BrightStor CA-Allocate receives control for these environments.

In the DEFINE environment, an EXIT CODE(0) statement causes volume selection to take place. Any exit code other than zero causes BrightStor CA-Allocate to return this value to the issuer of the catalog management request as the error return code. If a non zero exit code is specified, variables REASON and FAILMOD may also have been set before the EXIT statement to further qualify the reason for failing the allocation.

In the ACS environment, an EXIT CODE(0) causes any of the four SMS constructs that were set in the ASR and requested for this ACS invocation to be returned to the requestor of the ACS processing. An exit code other than 0 is returned to the requestor of the ACS processing, along with the ACS reason code that is set in variable &ACSRSN. None of the four SMS constructs set in the ASR is returned to the requestor. No matter what the exit code value, the values in the ACS return code variable (&ACSRC) and the ACS reason code variable (&ACSRSN) are returned to the requestor of the ACS processing, whether they are set in the ASR or left as they were upon entry to the ASR.

Note: When using BrightStor CA-Allocate to deny allocations of other program products, exiting with anything other than EXIT CODE(0) can cause unpredictable results.

The CALL Statement

```
CALL VDSEXIT?
CALL QUOTACHK
```

The CALL statement can invoke any one of ten possible User Exits (VDSEXIT0 through VDSEXIT9) in any ASR environment and the QUOTACHK internal routine in the QUOTA environment. For more information about User Exits, see the chapter "**User Exits**." For more information on QUOTACHK, see the section **VDSPROG ASR and the QUOTA** Environment in the chapter "**Installing**."

ASR Constants

Three kinds of constants can be specified in an Allocation Selection Routine: numbers, strings, and patterns.

Numbers

Numbers must be unsigned and less than or equal to 2147483647.

Strings

Strings can contain any character and are delimited by single quotation marks('). Two adjacent single quotation marks (') result in a single quotation mark in the string. Two adjacent ampersands (&&) result in an ampersand in the string. Strings can be up to 255 characters in length.

Note: An empty or null string is designated as follows: ' '.

Patterns

Patterns are formed by specifying non blank characters. The patterns are delimited by blanks, not quotes. Any ASR constant that is not numeric (that does not begin with digits 0 through 9), or a string (that is not contained within single quotation marks), is considered to be a pattern, even if it does not contain pattern-matching characters. A “pattern” without pattern-matching characters works like a string constant, but goes through a slightly longer code path.

Patterns can contain four kinds of special characters—the percent sign (%), a single asterisk (*), a double asterisk (**) and a question mark (?). They have the following meaning:

% - Takes the place of any single character during a comparison.

***** - Takes the place of any number of characters (including 0) during a comparison.

****** - Takes the place of any number of data set name nodes (including 0) during data set name comparisons.

‘?’ - Takes the place of any single numeric digit (in the range ‘0’ → ‘9’) during a comparison

For example:

SYS%.**.PROCLIB	Matches any PROCLIB that starts with SYS and one character.
SYS?.**.PARMLIB	Matches any PARMLIB that starts with SYS and one digit.
.*DMS*.	Matches any data set with DMS in it.
MVS*	Matches any volume that starts with MVS.
*RES	Matches any volume that ends with RES.
OOPS	Has no pattern-matching characters. This should have been a string constant (contained within single quotation marks), but works as if it were so formatted.

ASR Operators

ASR statements can include Boolean, comparison and subscript types of operators.

Boolean Operators

```
AND or &&       Designate a logical AND.
OR or |         Designate a logical OR.
```

All AND comparisons are evaluated before OR comparisons. Parentheses are used to override this normal order of evaluation. If multiple parentheses are used, the condition within the innermost set of parentheses is tested first, and evaluation proceeds outward.

For example:

```
IF (&USER='TIM' OR &USER='LAL') AND &HLQ='SYS1' THEN EXIT
```

Without the parentheses, the AND operation is done before the OR operation. In other words, the AND operator has a higher precedence than the OR operator.

Comparison Operators

EQ,=	equal
NE,Ô=	not equal
LT,<	less than
GT,>	greater than
LE,<=	less than or equal
GE,>=	greater than or equal

The abbreviations shown above or the mathematical symbols can be used interchangeably throughout the ASR.

Numeric comparisons are done unsigned. String comparisons are done in EBCDIC collating sequence. Pattern comparisons return either equal or not equal, because it is not possible to tell whether a pattern is greater than or less than a string. The same applies to list comparisons. (ANYVOL, ALLVOL, and STORGRP can be lists of items.) Patterns and filter lists may appear only on the right side of a comparison operator.

The following is not valid and is failed by the ASR compiler:

```
IF TEST* EQ &DSN(3) THEN...
```

The following is valid:

```
IF &DSN(3) EQ TEST* THEN...
```

Subscript Operator

Certain variables can take the subscript operator (numeric expression). A subscript that is out of range returns the first element of the subscripted variable. Subscript operators begin with the integer 1. The following variables can take the subscript operator:

ACCT_JOB	ACCT_STEP	CATDSN	DSN
DSND	DSNI	HSMDSN	LIKE
NEWNAME	QUOTAALLOCS	QUOTALIMITS	QUOTANAMES
QUOTAPERCENTS	QUOTASTATS	QUOTAWPERCENTS	REFDD
SECMODEL			

In the following example, the DSN variable contains the value 'TIM1.SAMPLE.DATASET' which results in the following:

```
&DSN(1) = 'TIM1'
&DSN(2) = 'SAMPLE'
```

```
&DSN(3) = 'DATASET'
```

ACCT_JOB contains job accounting information from a job's JOB card, and the ACCT_STEP variable contains step accounting information from the ACCT= parameter on the EXEC JCL statement for the step that is executing. Both of these can contain multiple pieces of data that can be picked up individually using the subscript operator.

If no subscript operator is used on the ACCT_JOB or ACCT_STEP variables, only the first piece of information is picked up (as if the subscript of 1 were used). However, when the CATDSN, DSN, DSND, and DSNI variables are used without a subscript operator, the entire data set name (no default subscript), is picked up.

Subscript operators cannot be used inside a WRITE statement. Instead, a user variable (C0 to C40) can be assigned the subscripted value, and that in turn can be used in a WRITE Statement. For details, see the section The WRITE Statement in this chapter.

ASR Functions

Two functions are available in ASR, the &SUBSTR and the Arithmetic functions.

The &SUBSTR Function

```
&SUBSTR(position[:length],string expression)
```

The &SUBSTR function returns a substring, beginning at position for the length if one is specified. Otherwise it returns the rest of the string expression out to the end. The first position is 1, not 0.

This example shows how a portion of a DSNAME can be set in a user variable.

```
IF &DSN = LABS.DMS???.LOADLIB THEN DO
   SET &C0 = &SUBSTR(4,&DSN(2)) /* GET THE ??? RELEASE NO.*/
   WRITE 'THE RELEASE IS &C0'
END
```

The following example shows how the first 3 characters of the job name can be placed in a user variable for later use in the ASR:

```
SET &C0 = &SUBSTR(1:3,&JOB)
IF &C0 = 'XXX' THEN DO
      ...
END
```

The &SUBSTR function can be used to look at any part of the character variables.

The &SUBSTR function cannot be used inside a WRITE statement. Instead, a user variable (C0 to C40) could be assigned the substring value, and that in turn could be used in a WRITE statement. For details, see the section The WRITE Statement in this chapter.

Arithmetic Expressions

Arithmetic expressions can be substituted for any numeric variable in the ASR. Constants, numeric variables, and operators may be used in expressions to set values within an ASR. The ASR language supports the following arithmetic operations:

+	Addition
-	Subtraction
*	Multiplication
/	Division (results in integer dividend of division)
%	Modulo (results in integer remainder of division)

Arithmetic operators must be preceded and followed by a space to distinguish them from pattern-matching operators.

The ASR language uses the basic rules of algebra when evaluating expressions and is governed by the following:

1. All expression results are calculated on the basis of operator precedence as defined below:
   - Multiplication, division, and modulo are carried out before addition and subtraction operations.
   - Operations of equal precedence are evaluated left to right.
   - Open parentheses “(“and close parentheses“)” are used to cluster operations. The operations within parentheses are evaluated first, starting with the innermost grouped functions when multiple sets of parentheses are used.
2. Precision in all intermediate and final results is subject to the following:
   - Signed 31-bit integer values that range from -21478368 through +21478367.
   - Significant digits are lost in overflow conditions that exceed the 31-bit range.
   - Decimal values are lost.

3. Division by zero results in zero.

The following equation demonstrates the rules of precedence and the precision of calculations:

$$7 \times 4 + 8 \div 3 - 2 \times 4 = ((7 \times 4) + (8 \div 3)) - (2 \times 4)$$
$$= (28 + 2) - 8$$
$$= 22$$

In the above equation, the intermediate calculations for the multiplication and division operations are performed before the addition and subtraction operations are completed. Also, note that the decimal portion of the intermediate result produced when 8 is divided by 3 is lost.

The following ASR code fragment demonstrates the use of ASR arithmetic.

```
IF &VAMENVIR = 'ALLOC' THEN                    /* IN THE ALLOC ENVIRONMENT      */
DO
   SET &PRIMARY = &PRIMARY * 3 / 2             /* INCREASE PRIMARY ALLOCATION   */
   SET &BLKSIZE = ((4096 / &LRECL) * &LRECL)   /* BY 50% AND ROUND THE BLKSIZE  */
END                                            /* TO THE NEAREST 4K BLKSIZE     */
```

Variables

TABLE 4-3 List of Variables

Variable (clicking on the variable will redirect you to the full description)	ACS	ALLOC	DEFINE	EOV	EOV_VSAM	EXTEND	OLD	RELEASE	RENAME	SCRATCH	SPACE
ABENDCOMP					I						
ACCT_JOB	I	I	I	I	I	I	I	I	I	I	I
ACCT_STEP	I	I	I	I	I	I	I	I	I	I	I
ACSENVIR	I										
ACSRC	I/O										
ACSRSN	I/O										
ALLVOL	I	I	I	I	I	I	I	I	I	I	I
ALLVOLD			I		I	I					
ALLVOLI			I		I	I					
ANYVOL	I	I	I	I	I	I	I		I	I	I
ANYVOLD			I		I	I					
ANYVOLI			I		I	I					
APPLIC	I	I	I	I	I	I	I		I	I	I
AVGBLK		I/O									I/O
AVGRECSIZE		I/O		I		I					I
AVGRECTYPE		I/O		I		I					I
BLKSIZE		I/O		I	I	I		I	I	I	I/O
BUFFERSPACE			I/O		I						
CANDIDATE_VOLUMES					I						
CAT_VOL_COUNT							I				
CATDSN			I/O		I						
CATHLQ			I		I						

Variable (clicking on the variable will redirect you to the full description)	ACS	ALLOC	DEFINE	EOV	EOV_VSAM	EXTEND	OLD	RELEASE	RENAME	SCRATCH	SPACE
CATLLQ			I		I						
CATNQUAL			I		I						
CATONDEFOK			I/O		I						
CISIZE			I/O		I	I					
CISIZED			I/O		I	I					
CISIZEI			I/O		I	I					
CMP3480		I/O									
CONTIG		I/O		I/O	I	I/O		I/O	I/O	I/O	I/O
C0 to C9	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O
C10 to C40	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O
CURRENT_PRIMARYD					I						
CURRENT_PRIMARYI					I						
CURRENT_SECONDARYD					I						
CURRENT_SECONDARYI					I						
DATACLAS	I/O	I	I	I[1]	I[1]	I[1]		I[1]	I[1]	I[1]	
DATASUFFIX			I/O								
DC_AVGBLK		I	I								
DC_AVGBLK_FLAG		I	I								
DC_AVGRECSIZE		I	I								
DC_AVGRECSIZE_FLAG		I	I								
DC_AVGRECTYPE		I	I								
DC_AVGRECTYPE_FLAG		I	I								
DC_CISIZED		I	I								
DC_CISIZED_FLAG		I	I								
DC_DIRBLOCKS		I	I								
DC_DIRBLOCKS_FLAG		I	I								

Variable (clicking on the variable will redirect you to the full description)	ACS	ALLOC	DEFINE	EOV	EOV_VSAM	EXTEND	OLD	RELEASE	RENAME	SCRATCH	SPACE
DC_DSNTYPE		I	I								
DC_DSNTYPE_FLAG		I	I								
DC_EXPDT		I	I								
DC_EXPDT_FLAG		I	I								
DC_FRSPCCA		I	I								
DC_FRSPCCA_FLAG		I	I								
DC_FRSPCCI		I	I								
DC_FRSPCCI_FLAG		I	I								
DC_IMBED		I	I								
DC_IMBED_FLAG		I	I								
DC_KEYLEN		I	I								
DC_KEYLEN_FLAG		I	I								
DC_KEYOFF		I	I								
DC_KEYOFF_FLAG		I	I								
DC_LMDDT		I	I								
DC_LMDTM		I	I								
DC_LMDUSR		I	I								
DC_LRECL		I	I								
DC_LRECL_FLAG		I	I								
DC_MAXVOL		I	I								
DC_MAXVOL_FLAG		I	I								
DC_PRIMARY		I	I								
DC_PRIMARY_FLAG		I	I								
DC_RECFM		I	I								
DC_RECFM_FLAG		I	I								
DC_RECORG		I	I								

Variable (clicking on the variable will redirect you to the full description)	ACS	ALLOC	DEFINE	EOV	EOV_VSAM	EXTEND	OLD	RELEASE	RENAME	SCRATCH	SPACE
DC_RECORG_FLAG		I	I								
DC_REPLICATE		I	I								
DC_REPLICATE_FLAG		I	I								
DC_RETPD		I	I								
DC_RETPD_FLAG		I	I								
DC_SECONDARY		I	I								
DC_SECONDARY_FLAG		I	I								
DC_XREGION		I	I								
DC_XREGION_FLAG		I	I								
DC_XSYSTEM		I	I								
DC_XSYSTEM_FLAG		I	I								
DD	I	I	I	I	I	I	I				
DDD			I		I	I					
DDI			I		I	I					
DEF_DATACLAS	I										
DEF_MGMTCLAS	I										
DEF_STORCLAS	I										
DEFER_TAPE_MOUNT		I/O									
DEVCLAS		I					I				
DIRBLOCKS		I/O				I					I
DISP		I		I		I	I				I
DISPA		I/O		I		I	I				I
DISPN		I/O		I		I	I				I
DMSGROUP	I	I	I	I	I	I	I	I	I	I	I
DMSUSR	I	I	I	I	I	I	I	I	I	I	I
DSN	I	I	I	I	I	I	I	I	I	I	I

Variable (clicking on the variable will redirect you to the full description)	ACS	ALLOC	DEFINE	EOV	EOV_VSAM	EXTEND	OLD	RELEASE	RENAME	SCRATCH	SPACE
DSND			I		I	I					
DSNI			I		I	I					
DSNTYPE	I										
DSORG	I	I/O	I	I	I	I		I	I	I	I
DSOWNER	I										
DSTYPE	I	I	I	I	I	I					I
EXPDT	I	I/O	I/O	I	I	I	I	I	I	I	I/O
EXPDT_CODED		I					I				
EXTENDCOMP						I					
EXTENDVOL						I					
EXTENTS		I	I	I		I		I	I	I	I
EXTENTSD					I	I					
EXTENTSI					I	I					
FAILIFNOVOLSEL		I/O	I/O								
FAILMOD			I/O								
FDR		I									
FILENUM		I/O									
FIXUNITMISMATCH							I				
GROUP	I	I	I	I	I	I	I	I	I	I	I
GUARANTEED_SPACE	I	I	I								
HLQ	I	I	I	I	I	I	I	I	I	I	I
HLQD			I		I	I					
HLQI			I		I	I					
HSMDSN		I									I
HSMFUNC	I	I	I								
HSMGROUP		I									I

Variable (clicking on the variable will redirect you to the full description)	ACS	ALLOC	DEFINE	EOV	EOV_VSAM	EXTEND	OLD	RELEASE	RENAME	SCRATCH	SPACE
HSMHLQ		I									I
HSMJOB		I									I
HSMLLQ		I									I
HSMNQUAL		I									I
HSMUSER		I									I
IFALREADYCAT	I	I	I	I	I	I	I	I	I	I	I
IFALREADYCATVOL	I	I	I	I	I	I	I	I	I	I	I
IMBED			I/O			I					
INDEXSEP			I/O								
INDEXSUFFIX			I/O								
JOB	I	I	I	I	I	I	I	I	I	I	I
JOBCATOK			I/O								
JOBCLASS		I	I	I	I	I	I	I	I	I	I
LABEL		I/O									
LARGEST_EXTCYL		I	I	I	I	I					
LARGEST_EXTCYL_NOT		I	I	I	I	I					
LARGEST_EXTMB		I	I	I	I	I					
LARGEST_EXTMB_NOT		I	I	I	I	I					
LARGEST_EXTTRK		I	I	I	I	I					
LARGEST_EXTTRK_NOT		I	I	I	I	I					
LIKE		I/O									
LIKEHLQ		I									
LIKELLQ		I									
LIKENQUAL		I									
LLQ	I	I	I	I	I	I	I	I	I	I	I
LLQD			I		I	I					

Variable (clicking on the variable will redirect you to the full description)	ACS	ALLOC	DEFINE	EOV	EOV_VSAM	EXTEND	OLD	RELEASE	RENAME	SCRATCH	SPACE
LLQI			I		I	I					
LRECL		I/O	I	I	I	I		I	I	I	I
LSPACE_FREESPACE		I	I	I	I	I					
LSPACE_PCTAFT_CYL		I	I	I	I	I					
LSPACE_PCTAFT_CYLNOT		I	I	I	I	I					
LSPACE_PCTAFT_MB		I	I	I	I	I					
LSPACE_PCTAFT_MBNOT		I	I	I	I	I					
LSPACE_PCTAFT_TRK		I	I	I	I	I					
LSPACE_PCTAFT_TRKNOT		I	I	I	I	I					
LSPACE_RETURN_CODE		I	I	I	I	I					
LSPACE_STORGRP		O	O	O	O						
LSPACE_VOLUME	O	O	O			O					
MAXSIZE	I	I		I		I					I
MGMTCLAS	I/O	I	I	I[1]	I[1]	I[1]		I[1]	I[1]	I[1]	I
MODULE	I	I	I	I	I	I		I	I	I	I
MSPOLICY	I										
MSPOOL	I										
MSVGP	I	I		I		I		I	I	I	I
MVOLSPC		I/O	I/O								
NEWNAME									I		
NEWNAMEHLQ									I		
NEWNAMELLQ									I		
NEWNAMENQUAL									I		
NKEYRANGE			I		I						
NQUAL	I	I	I	I	I	I	I	I	I	I	I
NQUALD			I		I	I					

Variable (clicking on the variable will redirect you to the full description)	ACS	ALLOC	DEFINE	EOV	EOV_VSAM	EXTEND	OLD	RELEASE	RENAME	SCRATCH	SPACE
NQUALI			I		I	I					
NTRKD					I						
NTRKI					I						
NUNIT		I/O	I/O				I/O				I
NVOL	I	I/O		I		I		I	I		I
NVOLD			I/O		I	I					
NVOLI			I/O		I	I					
N0 to N40	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O
OPTBLK		I									I
OWNERID			I/O		I						
PGM	I	I	I	I	I	I	I	I	I	I	I
PGMRNAME		I	I	I	I	I	I	I	I	I	I
POOLSUB		I/O	I/O	I/O	I/O						
PRIMARY	I	I/O	I/O	I	I	I					I/O
PRIMARYD			I/O		I/O	I					
PRIMARYI			I/O		I/O						
PROCSTEP		I	I	I	I	I	I	I	I	I	I
REASON			I/O								
RECFM		I/O		I		I		I	I	I	I
RECORG	I		I		I	I					
REFDD		I									
REFDDNQUAL		I									
REPLICATE			I/O			I					
RETPD	I	I/O	I/O	I	I	I	I	I	I	I	I/O
RLSE		I/O		I		I					I/O
SECMODEL		I									

Variable (clicking on the variable will redirect you to the full description)	ACS	ALLOC	DEFINE	EOV	EOV_VSAM	EXTEND	OLD	RELEASE	RENAME	SCRATCH	SPACE
SECMODG		I									
SECONDARY	I	I/O	I/O	I/O		I/O					I/O
SECONDARYD			I/O		I/O	I/O					
SECONDARYI			I/O		I/O	I/O					
SECONDARY_HAD_ZERO				I		I					
SECONDARY_ORIG	I	I		I		I					I
SHRINKTABLE			I/O		I						
SHRINKTASK			I		I						
SIZE	I										
SIZEKB	I										
SIZEMB	I										
SPACCVT		I/O	I/O								I/O
SPACTYPE	I	I/O	I/O	I		I					I/O
SPACTYPED			I/O		I	I					
SPACTYPEI			I/O		I	I					
STEPCATOK			I/O								
STEPNAME		I	I	I	I	I	I	I	I	I	I
STOP_NOT_CATLG2											I/O
STOP_NOT_CATLG2_NODE											I/O
STOP_NOT_CATLG2_RC											I
STORCLAS	I/O	I	I	I[1]	I[1]	I[1]		I[1]	I[1]	I[1]	I
STORGRP	I/O	I/O	I/O	I/O[1]	I/O[1]	I[1]		I[1]	I[1]	I[1]	
STORGRPD			O		O						
STORGRPI			O		O						
SYSID	I	I	I	I	I	I	I	I	I	I	I
TIME	I	I	I	I	I	I	I	I	I	I	I

Variable (clicking on the variable will redirect you to the full description)	ACS	ALLOC	DEFINE	EOV	EOV_VSAM	EXTEND	OLD	RELEASE	RENAME	SCRATCH	SPACE
TODAY	I	I	I	I	I	I	I	I	I	I	I
&UNIT	I	I/O	I	I	I		I/O				I
UNITAFF		I/O									
UNITD						I					
UNITI						I					
UNITMISMATCH							I				
UNITTYPE		I/O	I/O	I		I/O					I/O
UNITTYPED						I					
UNITTYPEI						I					
USER	I	I	I	I	I	I	I	I	I	I	I
VAMENVIR	I	I	I	I	I	I	I	I	I	I	I
VAMRLSE	I	I	I	I	I	I	I	I	I	I	I
VLCT		I/O	I								I
VOLSER						I					
VREFDDN		I/O									
VREFDSN		I/O									
VREFSTP		I/O									
WEEKDAY	I	I	I	I	I	I	I	I	I	I	I
WRITE_EOFM		I/O									

1 = DATACLAS, MGMTCLAS, STORCLAS, and STORGRP variables are available for INPUT in the EXTEND, RELEASE, RENAME, EOV, EOV_VSAM, AND SCRATCH Environments for SMS-Managed data sets only.

Description of ASR Variables

ABENDCOMP

Format type = character
Maximum size = 8
VSAM

Applicable during EOV_VSAM environment.

The ABENDCOMP variable identifies the VSAM component that has run out of candidate volumes from which to get space. It contains a descriptive name for the failing component of either DATA or INDEX.

The value is obtained from the VSAM AMB (Access Method Block).

Alias: AC

ACCT_JOB

Format type = character(sub)
Maximum size = 142
Non-VSAM or VSAM

The ACCT_JOB variable is a subscripted character variable. It contains the job accounting information from the Job Card.

If a Job Card looked like this:

```
//JOBNAME JOB (111,222,333),...
```

Then these ASR statements are true:

```
IF &ACCT_JOB = '111' AND           /* same as ACCT_JOB(1) */
   &ACCT_JOB(2) = '222' AND
   &ACCT_JOB(3) = '333' AND
   &ACCT_JOB(4) EQ '' THEN WRITE 'ALL TRUE'
```

ACCT_STEP

Format type = character(sub)
Maximum size = 142
Non-VSAM or VSAM

The ACCT_STEP variable is a subscripted character variable. It contains the step accounting fields from the ACCT= keyword on the EXEC JCL statement.

If an EXEC card looked like this:

```
//STEP EXEC PGM=IEFBR14,ACCT=(555,444)
```

Then these ASR statements are true:

```
IF &ACCT_STEP = '555' and            /* same as ACCT_STEP(1) */
&ACCT_STEP(2) = '444' AND
&ACCT_STEP(3) EQ '' THEN WRITE 'ALL TRUE'.
```

ACSENVIR

Format type = character
Maximum size = 8
Non-VSAM or VSAM
SMS-Managed or non-SMS-Managed

The ACSENVIR variable contains the ACS environment name that is running as follows:

'RECALL'	DFHSM RECALL's	'STORE'	OSREQ object store
'RECOVER'	DFHSM RECOVER's	'CHANGE'	OSREQ object change
'CONVERT'	data set convert in place	'CTRANS'	OSMC object class transition
'ALLOC'	new data set allocations	'other'	set by IBM ACS user exits.

ACSRC

Format type = numeric
Non-VSAM or VSAM
SMS-Managed or non-SMS-Managed

The ACSRC variable contains the return code value that is returned to the requestor of the ACS processing. The original value in this variable comes from the IBM ACS processing (if any). This value is returned regardless of the exit code value. The variable ACSRSN contains the associated reason code. The ACSRC variable can be used as the exit code value, as shown below:

```
EXIT CODE(&ACSRC)
```

ACSRSN

Format type = numeric
Non-VSAM or VSAM
SMS-Managed or non-SMS-Managed

The ACSRSN variable contains the reason code value that is returned to the requestor of the ACS processing. The original value in this variable comes from the IBM ACS processing (if any). This value is returned without regard for the exit code value. The variable ACSRC contains the associated return code.

ALLVOL

Format type = character(list)
Maximum size = 378 (54 volumes)
Non-VSAM or VSAM

The ALLVOL variable contains the list of volume serial numbers specified by the allocation requestor. In a comparison operation, all the volumes in the list are matched with the operand on the right side of the operator. If ALL of the volumes match, the comparison is true. When ALLVOL is used in a WRITE statement, only the first volume in the list is printed.

For ACS, this value is established by the requestor of the processing ACS.

For VSAM, this value is obtained from the Catalog Management base component control block.

For non-VSAM, if the requestor did not specify any volumes, the list is empty. To determine whether any volumes were specified by the requestor, use the following statement:

```
IF &ALLVOL = '' THEN WRITE 'NON-SPECIFIC REQUEST'
```

ALLVOLD

Format type = character(list)
Maximum size = 378 (54 volumes)
VSAM Data Component

The ALLVOLD variable contains the list of volume serial numbers specified by the allocation requestor. In a comparison operation, all the volumes in the list are matched with the operand on the right side of the operator. If ALL the volumes match, the comparison is true. When ALLVOLD is used in a WRITE statement, only the first volume in the list is printed.

This value is obtained from the data component catalog entry.

ALLVOLI

Format type = character(list)
Maximum size = 378 (54 volumes)
VSAM Index Component

The ALLVOLI variable contains the list of volume serial numbers specified by the allocation requestor. In a comparison operation, all the volumes in the list are matched with the operand on the right side of the operator. If ALL the volumes match, the comparison is true. When ALLVOLI is used in a WRITE statement, only the first volume in the list is printed.

This value is obtained from the index component catalog entry.

ANYVOL

Format type = character(list)
Maximum size = 378 (54 volumes)
Non-VSAM or VSAM

The ANYVOL variable contains the list of volume serial numbers specified by the allocation requestor. In a comparison operation, all the volumes in the list are matched with the operand on the right side of the operator. If ONE of the volumes matches, the comparison is true. The WRITE statement only prints the first volume in the list.

If the requestor did not specify any volumes, the list is empty. To determine whether any volumes were specified, use the following statement:

```
IF &ANYVOL='' THEN DO . . .
```

This condition is true if the volume count or unit count is greater than the number of volumes specified. For example:

```
//DD1 DD UNIT=(SYSDA,2),VOL=SER=(WRK001),SPACE=(TRK,(1))
```

Since two units are requested by the UNIT parameter but only one volume serial number is given, the first volume request is specific, and the second is nonspecific. Therefore, in the example, one volume would be equal to null ('').

For ACS, this value is established by the requestor of the processing ACS.

For VSAM, the list of volumes to be placed in the ANYVOL variable is retrieved from the base cluster catalog entry. Even if the user specifies a list of volumes for the cluster, catalog management often moves the list to the data component. Thus, the ANYVOL variable does not always have a list of the volumes specified. Use the ANYVOLD and ANYVOLI variables to reference the complete list of volumes in the DEFINE environment.

ANYVOLD

Format type = character(list)
Maximum size = 378 (54 volumes)
VSAM Data Component

The ANYVOLD variable contains the list of volume serial numbers specified by the allocation requestor. In a comparison operation, all the volumes in the list are matched with the operand on the right side of the operator. If ONE of the volumes matches, the comparison is true. When used in a WRITE statement, only the first volume in the list is printed.

The value of the ANYVOLD variable is obtained from the data component catalog entry.

ANYVOLI

Format type = character(list)
Maximum size = 378 (54 volumes)
VSAM Index Component

The ANYVOLI variable contains the list of volume serial numbers specified by the allocation requestor. In a comparison operation, all the volumes in the list are matched with the operand on the right side of the operator. If ONE of the volumes matches, the comparison is true. When used in a WRITE statement, only the first volume in the list is printed.

The value of the ANYVOLI variable is obtained from the index component catalog entry.

APPLIC

Format type = character
Maximum size = 8
Non-VSAM or VSAM

The APPLIC variable contains the RACF-assigned application identifier. If no identifier has been assigned, the APPLIC variable contains the null string''.

For ACS, this value is established by the requestor of the processing ACS.

AVGBLK

Format type = numeric
Non-VSAM
non-SMS-Managed

AVGBLK is the average block length (blklgth) used to allocate space if the allocation is done in BLOCKS.

AVGBLK can only be modified if SPACTYPE = BLK.

In the DD statement below, the average block length is 1600 (&AVGBLK = 1600). This is different from the block size, which is 800 in this DD (&BLKSIZE = 800).

```
//DD1    DD DISP=(NEW,CATLG,DELETE),
//       DSN=ISPKCH1.TEST.ONE,VOL=SER=SSL801,
//       UNIT=SYSALLDA,SPACE=(1600,(25,15)),
//       DCB=(LRECL=80,DSORG=PS,BLKSIZE=800).
```

AVGRECSIZE

Format type = numeric
Maximum size = 5
Non-VSAM
SMS-Managed

AVGRECSIZE (reclgth) is for allocations using the SMS AVGREC (average record size) parameters to specify the space to be allocated for a data set.

Note: AVGREC is a JCL parameter that is valid only when SMS is active.

For these allocations, the SPACTYPE variable contains a value of 'AVR'. The AVGRECSIZE variable contains the average record size to be used to calculate space for the data set.

This variable is obtained from the original allocation specifications. For JCL allocations, AVGRECSIZE is taken from the space type field of the SPACE parameter on the DD statement. In the following example, the AVGRECSIZE value is 200.

```
SPACE=(200,(1,5)),AVGREC=M
```

AVGRECSIZE can be accessed in the ALLOC and SPACE environment but can be changed only in the ALLOC environment. Valid values are from 1 to 65535.

AVGRECTYPE

Format type = character
Maximum size = 1
Non-VSAM
SMS-Managed

AVGRECTYPE is for allocations using the SMS AVGREC (average record size) parameters to specify the record request and space quantity. When an AVGREC allocation is made, the primary and secondary space values indicate the amount of space to be allocated in units, thousands, or millions of records. The AVGRECTYPE variable contains the value from the AVGREC parameter, which specifies the factor for primary and secondary space values.

AVGRECTYPE can be accessed in the ALLOC and SPACE environments, but may be changed only in the ALLOC environment. AVGRECTYPE can be changed only when AVGREC is also specified in the original allocation. Possible values are:

U Primary and Secondary request is in units (factor 1).

K Primary and Secondary request is in thousands (factor 1024).

M Primary and Secondary request is in millions (factor 1048576).

BLKSIZE

Format type = numeric
Non-VSAM

The BLKSIZE variable contains the block size, if any, of the data set. If no block size was assigned, the BLKSIZE variable contains zero. BLKSIZE can be set in the ALLOC or SPACE environments. If RECFM=F(B), LRECL must be available, in which case the specified BLKSIZE is rounded down to a multiple of the LRECL.

Note: This variable can be used to set the block size at allocation time, but a program could subsequently change the block size at OPEN time.

Note: For SMS-managed allocations, BrightStor CA-Allocate's ability to modify this variable is contingent upon having access to the appropriate system control blocks.

BUFFERSPACE

Format type = numeric
VSAM

The BUFFERSPACE variable is the value specified or defaulted in the cluster's catalog entry when it was defined or subsequently altered.

Alias: BUFSP

CANDIDATE_VOLUMES

Format type = character
Maximum size = 1
VSAM

The CANDIDATE_VOLUMES variable indicates whether the VSAM data set has any candidate volumes associated with it. Possible values are:

Y	Unused candidate volumes
N	No unused candidate volumes

This information can be used by the ASR to differentiate between VSAM data sets with candidate volumes and those without. Those without any unused candidate volumes will need to have the ASR direct BrightStor CA-Allocate to find a new volume for them to extend on to. For those with available candidate volumes, the only thing the ASR may need to do is adjust the primary or secondary space allocation amounts.

See the descriptions of the PRIMARYD, PRIMARYI, CURRENT_PRIMARYD, CURRENT_PRIMARYI, CURRENT_SECONDARYD, and CURRENT_SECONDARYI variables in this chapter and the section **EOV Support for VSAM Data** Sets with Unused Candidate Volumes in the chapter "**Concepts and Facilities**" in this guide.

Alias: CAND_VOLS

CAT_VOL_COUNT

Format type = numeric
Non-VSAM
NonSMS-Managed only

The CAT_VOL_COUNT variable contains the number of volumes in a data set's catalog entry. When the unit-count specified in JCL used to allocate an old data set is greater than the catalog volume count, IBM's Common Allocation DD Preparation services will select and allocate additional volumes (up to the unit count value) to the data set using the following criteria:

- For data sets with PERMANENT data set names, IBM services will only select any additional volumes from those mounted as STORAGE.
- For data sets with TEMPORARY data sets names, as in system generated ones, IBM services will only select any additional volumes from those mounted as PUBLIC or STORAGE.

When there are an insufficient number of STORAGE or PUBLIC volumes mounted, IBM services will issue the appropriate IEF244I and IEF877E WTOR messages requesting that additional volumes of the proper mount attribute be brought online.

For those data sets eligible for **OLD Environment** processing, the following ASR statements can be used to reduce unit-count (NUNIT variable) to less than or equal to the catalog volume count (CAT_VOL_COUNT variable):

```
IF &NUNIT > & CAT_VOL_COUNT THEN SET &NUNIT = 1
```

or –

```
IF &NUNIT > & CAT_VOL_COUNT THEN SET &NUNIT = & CAT_VOL_COUNT
```

If there are a sufficient number of applicable STORAGE or PUBLIC volumes mounted, and the unit- count specified in the JCL is greater than the actual number of volumes actually extended onto by the data set at the time it encounters an End-Of-Volume condition, then such allocations will not be considered eligible for EOV Recovery and the following messages will be issued.

```
V37000I TCTTIOT(#DEVICES) <> JFCB(#VOLUMES)
V37000I jobname,pgmname,module name,ddname *** BYPASSING DATASET ***
```

Then IBM's allocaton services will attempt to extend the data set onto one of the applicable STORAGE or PUBLIC volumes it had allocated to the data set earlier, during processing by its Common Allocation services.

To insure that CA-Allocate's Volume Selection Services are used when ever possible, do not code any unit-count values in the UNIT JCL parameter. For those data sets eligible for OLD Environment processing, the unit count can be reduced or eliminated as detailed above.

See the associated variable NUNIT.

Alias: CVC

CATDSN

Format type = character(sub)
Maximum size = 44
Default = 'Y'
VSAM

The CATDSN variable contains the data set name of the Catalog into which the data set being defined will be cataloged.

If the CATONDEFOK, STEPCATOK, or JOBCATOK variables change, the value of the CATDSN variable can also change.

Each time the CATDSN variable is referenced, the catalog to be used is determined based on the following order of precedence:

1. If the CATDSN variable has been set to a non-null value in the ASR, use that value as the catalog data set name.
2. If the variable CATONDEFOK is not equal to 'N' and the DEFINE does specify a catalog name, use that catalog name.
3. If the variable STEPCATOK is not equal to 'N' and a stepcat has been allocated to the jobstep, use the catalog to which the stepcat points.
4. If the variable JOBCATOK is not equal to 'N' and a jobcat has been allocated to the job, use the catalog to which the jobcat points.
5. A search is made for an alias record or a user catalog record that matches the high-level qualifier(s) of the data set being defined. Only operating systems that support Multi-Level Alias (MLA) would ever search for more than the high-level qualifier.

   Note: If a system uses MLA with a value greater than 1, BrightStor CA-Allocate must be brought up with the same number of levels of alias in effect as the operating system is using. See the description for **ALIASLVL=** Subcommand in the chapter "**Operation**."
6. If no catalog has been selected yet, use the System Master Catalog.

The CATDSN variable is a subscripted variable. Any node of the data set name may be accessed using the subscript operator. The maximum subscript is 22. Referring to a data set name node that does not exist returns the null string ''. To get the second level qualifier in user variable C0, use the following statement:

```
SET &C0 = &CATDSN(2)
```

If no subscript is used, the entire data set name is used.

See also the descriptions of the CATONDEFOK, STEPCATOK, JOBCATOK, CATNQUAL, CATHLQ, and CATLLQ variables.

CATHLQ

Format type = character
Maximum size = 8
VSAM

The CATHLQ variable contains the high-level qualifier of the Catalog data set name. CATHLQ is the same value as &CATDSN(1).

If the CATDSN, CATONDEFOK, STEPCATOK, or JOBCATOK variables change, the value of the CATHLQ variable can also change. See also the descriptions of the CATDSN, CATNQUAL, CATONDEFOK, STEPCATOK, JOBCATOK, and CATLLQ variables.

CATLLQ

Format type = character
Maximum size = 8
VSAM

The CATLLQ variable contains the low-level qualifier of the Catalog data set name. CATLLQ is the same value as &CATDSN(&CATNQUAL).

If the CATDSN, CATONDEFOK, STEPCATOK, or JOBCATOK variables change, the value of the CATLLQ variable can also change. See also the descriptions of the CATDSN, CATNQUAL, CATONDEFOK, STEPCATOK, JOBCATOK, and CATHLQ variables.

CATNQUAL

Format type = numeric
VSAM

The CATNQUAL variable is equal to the number of qualifiers (nodes) in the Catalog data set name. For example, the Catalog data set name ICFCAT.VISP805 causes CATNQUAL to be equal to 2.

If the CATDSN, CATONDEFOK, STEPCATOK, or JOBCATOK variables change, the value of the CATNQUAL variable can also change. See also the descriptions of the CATDSN, CATONDEFOK, STEPCATOK, JOBCATOK, CATHLQ, and CATLLQ variables.

CATONDEFOK

Format type = character
Maximum size = 1
Default = 'Y'
VSAM

The CATONDEFOK variable controls whether a catalog name specified on a DEFINE should be used (CATONDEFOK=Y) or should be disallowed (CATONDEFOK=N). If the define being processed does not have a catalog name specified, then changing this variable has no affect on the define.

CATONDEFOK can be used in conjunction with variables STEPCATOK and JOBCATOK to enforce the standard catalog structure that has been set up within a shop, as shown in this example:

```
SET &CATONDEFOK = 'N'
SET &STEPCATOK = 'N'
SET &JOBCATOK = 'N'
WRITE 'The proper catalog to use is &CATDSN'
```

If the CATONDEFOK variable changes, the value of variables CATDSN, CATNQUAL, CATHLQ, and CATLLQ can also change. See also the descriptions of the CATDSN, CATNQUAL, STEPCATOK, JOBCATOK, CATHLQ, and CATLLQ variables.

CISIZE

Format type = numeric
VSAM

The CISIZE variable contains the control interval size in bytes obtained from the Catalog Management base component control block.

CISIZED

Format type = numeric
VSAM

The CISIZED variable contains the control interval size in bytes obtained from the Catalog Management data component control block.

CISIZEI

Format type = numeric
VSAM

The CISIZEI variable contains the control interval size in bytes obtained from the Catalog Management index component control block.

CMP3480

Format type = character
Maximum size = 1
Non-VSAM

Data compression was added as an option to the configuration of 3480/3490 tape devices. The CMP3480 variable represents the type of data compression specified for the current allocation request. It is applicable to new tape data sets only. Possible values are:

Y	TRTCH=COMP was specified
N	TRTCH=NOCOMP was specified
‘ ‘	no TRTCH parameter was specified

Possible output values are:

Y	turn-on data compression
N	turn-off data compression

For TAPE-to-DISK device type conversions, data compression is automatically turned off. For DISK-to-TAPE device type conversions, no default value for the CMP3480 variable is assumed. A detailed explanation of why different actions are taken is found in the section **Device-Type Conversions** in the chapter ”**Concepts and Facilities**.”

CONTIG

Format type = character
Maximum size = 1
Non-VSAM

The CONTIG variable contains 'Y' if the requestor specified CONTIG in the JCL SPACE parameter. Otherwise it contains 'N'.

```
SET &CONTIG='Y' /* FORCE CONTIGUOUS ALLOCATION */
```

Note: For SMS-managed allocations, BrightStor CA-Allocate's ability to modify this variable is contingent upon having access to the appropriate system control blocks.

CURRENT_PRIMARYD **Format type = numeric**
VSAM Data Component

The CURRENT_PRIMARYD variable contains the current primary quantity obtained from the applicable system control block (Access Method Block or AMB) associated with the VSAM Data Component in units specified by the SPACTYPED variable. When the VSAM cluster opens, this value is obtained from the data component catalog entry containing the data component creation information. If an EOV condition is encountered after the data set is opened, and BrightStor CA-Allocate was already used to change this space amount for example, during an EXTEND Environment invocation, then the value reflects the new space amount.

It is this current primary allocation amount that BrightStor CA-Allocate (NonSMS-managed only) or dfSMS (SMS-managed only) uses during the applicable volume selection processes. This is the amount used by VSAM when it makes the first extent on the new volume selected by either BrightStor CA-Allocate or dfSMS.

To change this amount, 'SET' the &PRIMARYD variable to the new allocation quantity. An example of this is shown in Figure 2-12.

See the descriptions of the CANDIDATE_VOLUMES, PRIMARYD, PRIMARYI, CURRENT_PRIMARYI, CURRENT_SECONDARYD, and CURRENT_SECONDARYI variables in this chapter and the section **EOV Support for VSAM Data** Sets with Unused Candidate Volumes in the chapter "**Concepts and Facilities**" in this guide.

Note: If the "Additional Volume Amount = SECONDARY" Data Class attribute is associated with the cluster, then IBM services will have changed this space amount to reflect what secondary amount the data component was originally allocated with.

Alias: CURR_PDC or CPDC

CURRENT_PRIMARYI **Format type = numeric**
VSAM Index Component

The CURRENT_PRIMARYI variable contains the current primary quantity obtained from the applicable system control block (Access Method Block or AMB) associated with the VSAM Index Component in units specified by the SPACTYPEI variable. When the VSAM cluster opens, this value is obtained from the index component catalog entry containing the index component creation information. If an EOV condition is encountered after the data set is opened, and BrightStor CA-Allocate was already used to change this space amount, for example, during an EXTEND Environment invocation, then the value reflects the new space amount.

It is this current primary allocation amount that BrightStor CA-Allocate (NonSMS-managed only) or dfSMS (SMS-managed only) uses during the applicable volume selection processes. This is the amount used by VSAM when it makes the first extent on the new volume selected by either BrightStor CA-Allocate or dfSMS.

To change this amount, 'SET' the &PRIMARYI variable to the new allocation quantity. An example of this is shown in Figure 2-12.

See the descriptions of the CANDIDATE_VOLUMES, PRIMARYD, PRIMARYI, CURRENT_PRIMARYI, CURRENT_SECONDARYD, and CURRENT_SECONDARYI variables in this chapter and the section **EOV Support for VSAM Data** Sets with Unused Candidate Volumes in the chapter "**Concepts and Facilities**" in this guide.

Note: If the "Additional Volume Amount = SECONDARY" Data Class attribute is associated with the cluster, then IBM services will have changed this space amount to reflect what secondary amount the index component was originally allocated with.

Alias: CURR_PIC or CPIC

CURRENT_SECONDARYD

Format type = numeric
VSAM Data Component

The CURRENT_SECONDARYD variable contains the current secondary quantity obtained from the applicable system control block (Access Method Block, or AMB) associated with the VSAM data component in units specified by the SPACTYPED variable. When the VSAM cluster opens, this value is obtained from the data component catalog entry containing the index component creation information. If an EOV condition is encountered after the data set is opened, and BrightStor CA-Allocate was already used to change this space amount, for example, during an EXTEND Environment invocation, then the value reflects the new space amount.

This is the allocation amount that will be used by VSAM when it makes the makes the second, as well as any subsequent, extent on the new volume selected by either BrightStor CA-Allocate or dfSMS.

To change this amount, 'SET' the &SECONDARYD variable to the new allocation quantity. An example of this is shown in Figure 2-12.

See the descriptions of the CANDIDATE_VOLUMES, PRIMARYD, PRIMARYI, CURRENT_PRIMARYI, CURRENT_SECONDARYD, and CURRENT_SECONDARYI variables in this chapter and the section **EOV Support for VSAM Data** Sets with Unused Candidate Volumes in the chapter "**Concepts and Facilities**" in this guide.

Alias: CURR_SDC or CSDC

CURRENT_SECONDARYI

Format type = numeric
VSAM Index Component

The CURRENT_SECONDARYI variable contains the current secondary quantity obtained from the applicable system control block (Access Method Block, or AMB) associated with the VSAM index component in units specified by the SPACTYPEI variable. When the VSAM cluster opens, this value is obtained from the index component catalog entry containing the index component creation information. If an EOV condition is encountered after the data set is opened, and BrightStor CA-Allocate was already used to change this space amount, for example, during an EXTEND Environment invocation, then the value reflects the new space amount.

This is the allocation amount that will be used by VSAM when it makes the makes the second, as well as any subsequent, extent on the new volume selected by either BrightStor CA-Allocate or dfSMS.

To change this amount, 'SET' the &SECONDARYI variable to the new allocation quantity. An example of this is shown in Figure 2-12.

See the descriptions of the CANDIDATE_VOLUMES, PRIMARYD, PRIMARYI, CURRENT_PRIMARYI, CURRENT_SECONDARYD, and CURRENT_SECONDARYI variables in this chapter and the section **EOV Support for VSAM Data** Sets with Unused Candidate Volumes in the chapter "**Concepts and Facilities**" in this guide.

Alias: CURR_SIC or CSIC

C0 to C9

Format type = character
Maximum size = 44
Non-VSAM and VSAM

The C0 through C9 variables are user-character variables. These variables are used to pass character information into and out of a User Exit and can be used as work variables as needed. For example,

```
SET &C0 = &XMODE
CALL VDSEXIT6 /* GET OUT IF THIS IS A STARTED TASK */
IF &C1 = 'STOP' THEN EXIT /* C1 WAS SET BY THE EXIT */
```

In the above example, the User Exit can check for TSO, BATCH, or TASK and copy 'STOP' into C1 if the execution mode is TASK. The following is equivalent:

```
IF &XMODE='TASK' THEN EXIT
```

C10 to C40

Format type = character
Maximum size = 254
Non-VSAM and VSAM

The C10 through C40 variables are identical to the C0 through C9 variables, except that their maximum size is 254.

DATACLAS

Format type = character
Maximum size = 8
Non-VSAM and VSAM
SMS-Managed or non-SMS-Managed

The DATACLAS variable contains the SMS data class construct name. DATACLAS is set from the original user request and any IBM ACS processing. In the ACS environment, the value can be set to any valid data class name or removed by setting it to null.

In DEFINE and SPACE environments, this information may not always be available.

The Data Class name may not always be available in the DEFINE, SCRATCH, and SPACE environments or in the QUOTA environment when QUOTAFUNC=SCRATCH.

Alias: DATACLASS or DC

DATASUFFIX

Format type = character
Maximum size = 8
VSAM Data Component

The DATASUFFIX variable is used to generate the data component name based on the base cluster name. The value of variable DATASUFFIX is appended to the end of the base cluster name to create the data component name. This is typically used to make the data component name end in ".DATA" or another installation standard.

The initial value of DATASUFFIX is a null string. If logic in the ASR sets DATASUFFIX to a non-null value, the data component name is generated or replaced when the "EXIT CODE(0)" ASR statement is executed. Note that the value of variable DSND is not altered when DATASUFFIX is assigned a value.

For example, if the base cluster name is MY.VSAM.FILE, the ASR statement shown below results in the data component name being set to MY.VSAM.FILE.DATA when the "EXIT CODE(0)" ASR statement is executed.

```
SET &DATASUFFIX = 'DATA'
```

DC_AVGBLK

Format type = numeric
Non-VSAM and VSAM
SMS-Managed or Non-SMS-Managed

The DC_AVGBLK variable contains the average block size of the associated Data Class if and only if the related variable DC_AVGBLK_FLAG = 'Y'. The DC_AVGBLK_FLAG is 'Y' only when the data set has a valid SMS Data Class associated with it and the space specified on the Data Class was an average blocks request.

See the associated variables AVGBLK, DC_AVGBLK_FLAG, and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCAVBS

DC_AVGBLK_FLAG

Format type = character
Maximum size = 1
Non-VSAM and VSAM
SMS-Managed or Non-SMS-Managed

The DC_AVGBLK_FLAG variable is a "yes" or "no" flag indicating whether or not the current allocation has a valid SMS Data Class associated with it and the space specified on the Data Class was an average blocks request. Possible values are:

Y	Data Class information available
N	No associated Data Class information

See the associated variables AVGBLK, DC_AVGBLK_FLAG, and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCAVBSF

DC_AVGRECSIZE

Format type = numeric
Non-VSAM and VSAM
SMS-Managed or Non-SMS-Managed

The DC_AVGRECSIZE variable contains the average record size of the associated Data Class if and only if the related variable DC_AVGRECSIZE_FLAG = 'Y'. The DC_AVGRECSIZE_FLAG is 'Y' only when the data set has a valid SMS Data Class associated with it and the space specified on the Data Class was an average records request.

See the associated variables AVGRECSIZE, AVGRECTYPE, DC_AVGRECTYPE, DC_AVGRECSIZE_FLAG, and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCAVRS

DC_AVGRECSIZE_FLAG

Format type = character
Maximum size = 1
Default = 'N'
Non-VSAM and VSAM
SMS-Managed or Non-SMS-Managed

The DC_AVGRECSIZE_FLAG variable is a "yes" or "no" flag indicating whether or not the current allocation has a valid SMS Data Class associated with it and the space specified on the Data Class was an average records request. Possible values are:

Y	Data Class information available
N	No associated Data Class information

See the associated variables AVGRECSIZE, AVGRECTYPE, DC_AVGRECTYPE, DC_AVGRECSIZE, and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCAVRSF

DC_AVGRECTYPE

Format type = character
Maximum size = 1
Non-VSAM or VSAM
SMS-Managed or Non-SMS-Managed

The DC_AVGRECTYPE variable contains the average record type of the associated Data Class if and only if the related variable DC_AVGRECTYPE_FLAG = 'Y'. The DC_AVGRECTYPE_FLAG is 'Y' only when the data set has a valid SMS Data Class associated with it and the space specified on the Data Class was an average records request.

See the associated variables AVGRECSIZE, AVGRECTYPE, DC_AVGRECSIZE, DC_AVGRECTYPE_FLAG, and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCAVRT

DC_AVGRECTYPE_FLAG

Format type = character
Maximum size = 1
Default = 'N'
Non-VSAM or VSAM
SMS-Managed or Non-SMS-Managed

The DC_AVGRECTYPE_FLAG variable is a "yes" or "no" flag indicating whether or not the current allocation has a valid SMS Data Class associated with it and the space specified on the Data Class was an average records request. Possible values are:

Y	Data Class information available
N	No associated Data Class information

See the associated variables AVGRECSIZE, AVGRECTYPE, DC_AVGRECSIZE, DC_AVGRECTYPE and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCAVRTF

DC_CISIZED **Format type = numeric**
VSAM
SMS-Managed or Non-SMS-Managed

The DC_CISIZED variable contains the CI size for the data component of the associated Data Class if and only if the related variable DC_CISIZED_FLAG = 'Y'. The DC_CISIZED_FLAG is 'Y' only when the data set has a valid SMS Data Class associated with it and the CI size for the data component was specified on the Data Class.

See the associated variables CISIZED, DC_CISIZED_FLAG, and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCCSZD

DC_CISIZED_FLAG **Format type = character**
Maximum size = 1
VSAM
SMS-Managed or Non-SMS-Managed

The DC_CISIZED_FLAG variable is a "yes" or "no" flag indicating whether or not the current allocation has a valid SMS Data Class associated with it and the control interval size of the data component was specified on the Data Class. Possible values are:

Y	Data Class information available
N	No associated Data Class information

See the associated variables CISIZED, DC_CISIZED, and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCCSZDF

DC_DIRBLOCKS

Format type = numeric
Non-VSAM
SMS-Managed or Non-SMS-Managed

The DC_DIRBLOCKS variable contains the directory blocks of the associated Data Class if and only if the related variable DC_DIRBLOCKS_FLAG = 'Y'. The DC_DIRBLOCKS_FLAG is 'Y' only when the data set has a valid SMS Data Class associated with it and the space specified on the Data Class included a directory blocks amount.

See the associated variables DIRBLOCKS, DC_DIRBLOCKS_FLAG, and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCDIRB

DC_DIRBLOCKS_FLAG

Format type = character
Maximum size = 1
Non-VSAM
SMS-Managed or Non-SMS-Managed

The DC_DIRBLOCKS_FLAG variable is a "yes" or "no" flag indicating whether or not the current allocation has a valid SMS Data Class associated with it and the space specified on the Data Class included a directory blocks amount. Possible values are:

Y	Data Class information available
N	No associated Data Class information

See the associated variables DIRBLOCKS, DC_DIRBLOCKS, and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCDIRBF

DC_DSNTYPE

Format type = character
Maximum size = 7
Non-VSAM
SMS-Managed or Non-SMS-Managed

The DC_DSNTYPE variable contains the data set type of the associated Data Class if and only if the related variable DC_DSNTYPE_FLAG = 'Y'. The DC_DSNTYPE_FLAG is 'Y' only when the data set has a valid SMS Data Class associated with it and the data set name type was specified on the Data Class.

See the associated variables DSNTYPE, DC_DSNTYPE_FLAG, and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCDSNT

DC_DSNTYPE_FLAG

Format type = character
Maximum size = 1
Default = 'N'
Non-VSAM
SMS-Managed or Non-SMS-Managed

The DC_DSNTYPE_FLAG variable is a "yes" or "no" flag indicating whether or not the current allocation has a valid SMS Data Class associated with it and the data set name type was specified on the Data Class. Possible values are:

Y	Data Class information available
N	No associated Data Class information

See the associated variables DSNTYPE, DC_DSNTYPE, and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCDSNTF

DC_EXPDT

Format type = numeric
Non-VSAM or VSAM
SMS-Managed or Non-SMS-Managed

The DC_EXPDT variable contains the expiration date of the associated Data Class if and only if the related variable DC_EXPDT_FLAG = 'Y'. The DC_EXPDT_FLAG is 'Y' only when the data set has a valid SMS Data Class associated with it and the expiration was specified on the Data Class.

See the associated variables EXPDT, DC_EXPDT_FLAG, and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCEXPD

DC_EXPDT_FLAG

Format type = character
Maximum size = 1
Default = 'N'
Non-VSAM or VSAM
SMS-Managed or Non-SMS-Managed

The DC_EXPDT_FLAG variable is a "yes" or "no" flag indicating whether or not the current allocation has a valid SMS Data Class associated with it and an expiration date was specified on the Data Class. Possible values are:

Y	Data Class information available
N	No associated Data Class information

See the associated variables EXPDT, DC_EXPDT, and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCEXPDF

DC_FRSPCCA **Format type = numeric**
VSAM
SMS-Managed or Non-SMS-Managed

The DC_FRSPCCA variable contains the percent free space for CA of the associated Data Class if and only if the related variable DC_FRSPCCA_FLAG = 'Y'. The DC_FRSPCCA_FLAG is 'Y' only when the data set has a valid SMS Data Class associated with it and the percent free space for CA was specified on the Data Class.

See the associated variables DC_FRSPCCA_FLAG and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCFSCA

DC_FRSPCCA_FLAG **Format type = character**
Maximum size = 1
Default = 'N'
VSAM
SMS-Managed or Non-SMS-Managed

The DC_FRSPCCA_FLAG variable is a "yes" or "no" flag indicating whether or not the current allocation has a valid SMS Data Class associated with it and the percent free space for the control area of the VSAM component was specified on the Data Class. Possible values are:

Y	Data Class information available
N	No associated Data Class information

See the associated variables DC_FRSPCCA and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCFSCAF

DC_FRSPCCI **Format type = numeric**
VSAM
SMS-Managed or Non-SMS-Managed

The DC_FRSPCCI variable contains the percent free space for CI of the associated Data Class if and only if the related variable DC_FRSPCCI_FLAG = 'Y'. The DC_FRSPCCI_FLAG is 'Y' only when the data set has a valid SMS Data Class associated with it and the percent free space for CI was specified on the Data Class.

See the associated variables DC_FRSPCCI_FLAG and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCFSCI

DC_FRSPCCI_FLAG **Format type = character**
Maximum size = 1
Default = 'N'
VSAM
SMS-Managed or Non-SMS-Managed

The DC_FRSPCCI_FLAG variable is a "yes" or "no" flag indicating whether or not the current allocation has a valid SMS Data Class associated with it and the percent free space for the control interval of the VSAM component was specified on the Data Class. Possible values are:

Y	Data Class information available
N	No associated Data Class information

See the associated variables DC_FRSPCCI and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCFSCIF

DC_IMBED

Format type = character
Maximum size = 1
VSAM
SMS-Managed or Non-SMS-Managed

The DC_IMBED variable contains the IMBED indicator of the associated Data Class if and only if the related variable DC_IMBED_FLAG = 'Y'. The DC_IMBED_FLAG is 'Y' only when the data set has a valid SMS Data Class associated with it and the IMBED indicator was set to yes on the associated Data Class.

See the associated variables DC_IMBED_FLAG and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCIMBD

DC_IMBED_FLAG

Format type = character
Maximum size = 1
Default = 'N'
VSAM
SMS-Managed or Non-SMS-Managed

The DC_IMBED_FLAG variable is a "yes" or "no" flag indicating whether or not the current allocation has a valid SMS Data Class associated with it and imbed was specified on the Data Class. Possible values are:

Y	Data Class information available
N	No associated Data Class information

See the associated variables DC_IMBED and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCIMBDF

DC_KEYLEN

Format type = numeric
VSAM
SMS-Managed or Non-SMS-Managed

The DC_KEYLEN variable contains the key length of the associated Data Class if and only if the related variable DC_KEYLEN_FLAG = 'Y'. The DC_KEYLEN_FLAG is 'Y' only when the data set has a valid SMS Data Class associated with it and the key length was specified on the Data Class.

See the associated variables DC_KEYLEN_FLAG and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCKEYL

DC_KEYLEN_FLAG

Format type = character
Maximum size = 1
Default = 'N'
VSAM
SMS-Managed or Non-SMS-Managed

The DC_KEYLEN_FLAG variable is a "yes" or "no" flag indicating whether or not the current allocation has a valid SMS Data Class associated with it and the key length was specified on the Data Class. Possible values are:

Y	Data Class information available
N	No associated Data Class information

See the associated variables DC_KEYLEN and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCKEYLF

DC_KEYOFF

Format type = numeric
VSAM
SMS-Managed or Non-SMS-Managed

The DC_KEYOFF variable contains the key offset of the associated Data Class if and only if the related variable DC_KEYOFF_FLAG = 'Y'. The DC_KEYOFF_FLAG is 'Y' only when the data set has a valid SMS Data Class associated with it and the key offset was specified on the Data Class.

See the associated variables DC_KEYLEN_FLAG and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCKEYO

DC_KEYOFF_FLAG

Format type = character
Maximum size = 1
Default = 'N'
VSAM
SMS-Managed or Non-SMS-Managed

The DC_KEYOFF_FLAG variable is a "yes" or "no" flag indicating whether or not the current allocation has a valid SMS Data Class associated with it and the key offset was specified on the Data Class. Possible values are:

Y	Data Class information available
N	No associated Data Class information

See the associated variables DC_KEYOFF and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCKEYOF

DC_LMDDT **Format type = character**
Maximum size = 10
SMS-Managed or Non-SMS-Managed

The DC_LMDDT variable contains the date of the last modification to the associated Data Class.

See the associated variables DC_LMDTM and DC_LMDUSR. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCLMDD

DC_LMDTM **Format type = character**
Maximum size = 8
SMS-Managed or Non-SMS-Managed

The DC_LMDTM variable contains the time of the last modification to the associated Data Class.

See the associated variables DC_LMDDT and DC_LMDUSR. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCLMDT

DC_LMDUSR **Format type = character**
Maximum size = 8
SMS-Managed or Non-SMS-Managed

The DC_LMDUSR variable contains the userid of the user that last modified the associated Data Class.

See the associated variables DC_LMDDT and DC_LMDTM. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCLMDU

DC_LRECL

Format type = numeric
Non-VSAM
SMS-Managed or Non-SMS-Managed

The DC_LRECL variable contains the logical record length of the associated Data Class if and only if the related variable DC_LRECL_FLAG = 'Y'. The DC_LRECL_FLAG is 'Y' only when the data set has a valid SMS Data Class associated with it and the logical record length was specified on the Data Class.

See the associated variables LRECL, DC_LRECL_FLAG, and DATACLAS. Also section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCLRCL

DC_LRECL_FLAG

Format type = character
Maximum size = 1
Default = 'N'
Non-VSAM
SMS-Managed or Non-SMS-Managed

The DC_LRECL_FLAG variable is a "yes" or "no" flag indicating whether or not the current allocation has a valid SMS Data Class associated with it and the logical record length was specified on the Data Class. Possible values are:

Y	Data Class information available
N	No associated Data Class information

See the associated variables LRECL, DC_LRECL, and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCLRCLF

DC_MAXVOL

Format type = numeric
Non-VSAM or VSAM
SMS-Managed or Non-SMS-Managed

The DC_MAXVOL variable contains the maximum number of volumes that a data set should span for the associated Data Class if and only if the related variable DC_MAXVOL_FLAG = 'Y'. The DC_MAXVOL_FLAG is 'Y' only when the data set has a valid SMS Data Class associated with it and the volume count was specified on the Data Class.

See the associated variables NUNIT, NVOL, VLCT, DC_MAXVOL_FLAG, and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCMAXV

DC_MAXVOL_FLAG

Format type = character
Maximum size = 1
Default = 'N'
Non-VSAM or VSAM
SMS-Managed or Non-SMS-Managed

The DC_MAXVOL_FLAG variable is a "yes" or "no" flag indicating whether or not the current allocation has a valid SMS Data Class associated with it and the volume count was specified on the Data Class. Possible values are:

Y	Data Class information available
N	No associated Data Class information

See the associated variables NUNIT, NVOL, VLCT, DC_MAXVOL and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCMAXVF

DC_PRIMARY

Format type = numeric
Non-VSAM or VSAM
SMS-Managed or Non-SMS-Managed

The DC_PRIMARY variable contains the primary space request of the associated Data Class if and only if the related variable DC_PRIMARY_FLAG = 'Y'. The DC_PRIMARY_FLAG is 'Y' only when the data set has a valid SMS Data Class associated with it and the primary space was specified on the Data Class.

See the associated variables PRIMARY, SECONDARY, DC_SECONDARY, DC_PRIMARY_FLAG, and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCPRIM

DC_PRIMARY_FLAG

Format type = character
Maximum size = 1
Default = 'N'
Non-VSAM or VSAM
SMS-Managed or Non-SMS-Managed

The DC_PRIMARY_FLAG variable is a "yes" or "no" flag indicating whether or not the current allocation has a valid SMS Data Class associated with it and the primary space amount was specified on the Data Class. Possible values are:

Y	Data Class information available
N	No associated Data Class information

See the associated variables PRIMARY, SECONDARY, DC_SECONDARY, DC_PRIMARY, and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCPRIMF

DC_RECFM

Format type = character
Maximum size = 4
Non-VSAM or VSAM
SMS-Managed or Non-SMS-Managed

The DC_RECFM variable contains the record format of the associated Data Class if and only if the related variable DC_RECFM_FLAG = 'Y'. The DC_RECFM_FLAG is 'Y' only when the data set has a valid SMS Data Class associated with it and the record format was specified on the Data Class.

See the associated variables RECFM, DC_RECFM_FLAG, and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCRCFM

DC_RECFM_FLAG

Format type = character
Maximum size = 1
Default = 'N'
Non-VSAM
SMS-Managed or Non-SMS-Managed

The DC_RECFM_FLAG variable is a "yes" or "no" flag indicating whether or not the current allocation has a valid SMS Data Class associated with it and the record format was specified on the Data Class. Possible values are:

Y	Data Class information available
N	No associated Data Class information

See the associated variables RECFM, DC_RECFM, and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCRCFMF

DC_RECORG

Format type = character
Maximum size = 2
VSAM JCL
SMS-Managed or Non-SMS-Managed

The DC_RECORG variable contains the record organization of the associated Data Class if and only if the related variable DC_RECORG_FLAG = 'Y'. The DC_RECORG_FLAG is 'Y' only when the data set has a valid SMS Data Class associated with it and the record organization was specified on the Data Class.

See the associated variables RECORG, DC_RECORG_FLAG, and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCRCO

DC_RECORG_FLAG

Format type = character
Maximum size = 1
Default = 'N'
SMS-Managed or Non-SMS-Managed

The DC_RECORG_FLAG variable is a "yes" or "no" flag indicating whether or not the current allocation has a valid SMS Data Class associated with it and the record organization was specified on the Data Class. Possible values are:

Y	Data Class information available
N	No associated Data Class information

See the associated variables RECORG, DC_RECORG, and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCRCOF

DC_REPLICATE

Format type = character
Maximum size = 1
VSAM
SMS-Managed or Non-SMS-Managed

The DC_REPLICATE variable contains the replicate indicator of the associated Data Class if and only if the related variable DC_REPLICATE_FLAG = 'Y'. The DC_REPLICATE_FLAG is 'Y' only when the data set has a valid SMS Data Class associated with it and replicate was specified on the Data Class.

See the associated variables DC_REPLICATE_FLAG and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCREPL

DC_REPLICATE_FLAG

Format type = character
Maximum size = 1
Default = 'N'
VSAM
SMS-Managed or Non-SMS-Managed

The DC_REPLICATE_FLAG variable is a "yes" or "no" flag indicating whether or not the current allocation has a valid SMS Data Class associated with it and replicate was specified on the Data Class. Possible values are:

Y	Data Class information available
N	No associated Data Class information

See the associated variables DC_REPLICATE and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCREPLF

DC_RETPD

Format type = numeric
Non-VSAM
SMS-Managed or Non-SMS-Managed

The DC_RETPD variable contains the retention period of the associated Data Class if and only if the related variable DC_RETPD_FLAG = 'Y'. The DC_RETPD_FLAG is 'Y' only when the data set has a valid SMS Data Class associated with it and the retention period was specified on the Data Class.

See the associated variables RETPD, DC_RETPD_FLAG, and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCRTPD

DC_RETPD_FLAG

Format type = character
Maximum size = 1
Default = 'N'
Non-VSAM or VSAM
SMS-Managed or Non-SMS-Managed

The DC_RETPD_FLAG variable is a "yes" or "no" flag indicating whether or not the current allocation has a valid SMS Data Class associated with it and retention period was specified on the Data Class. Possible values are:

Y	Data Class information available
N	No associated Data Class information

See the associated variables RETPD, DC_RETPD, and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCRTPDF

DC_SECONDARY

Format type = numeric
Non-VSAM or VSAM
SMS-Managed or Non-SMS-Managed

The DC_SECONDARY variable contains the secondary space request of the associated Data Class if and only if the related variable DC_SECONDARY_FLAG = 'Y'. The DC_SECONDARY_FLAG is 'Y' only when the data set has a valid SMS Data Class associated with it and the secondary space was specified on the Data Class.

See the associated variables PRIMARY, SECONDARY, DC_PRIMARY, DC_SECONDARY_FLAG, and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCSEC

DC_SECONDARY_FLAG

Format type = character
Maximum size = 1
Default = 'N'
Non-VSAM or VSAM
SMS-Managed or Non-SMS-Managed

The DC_SECONDARY_FLAG variable is a "yes" or "no" flag indicating whether or not the current allocation has a valid SMS Data Class associated with it and the secondary space amount was specified on the Data Class. Possible values are:

Y	Data Class information available
N	No associated Data Class information

See the associated variables PRIMARY, SECONDARY, DC_PRIMARY, DC_SECONDARY, and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCSECF

DC_XREGION **Format type = numeric**
VSAM
SMS-Managed or Non-SMS-Managed

The DC_XREGION variable contains the share option for region of the associated Data Class if and only if the related variable DC_XREGION_FLAG = 'Y'. The DC_XREGION_FLAG is 'Y' only when the data set has a valid SMS Data Class associated with it and the share option for region was specified on the Data Class.

See the associated variables DC_XREGION_FLAG and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCSHXR

DC_XREGION_FLAG **Format type = character**
Maximum size = 1
Default = 'N'
VSAM
SMS-Managed or Non-SMS-Managed

The DC_XREGION_FLAG variable is a "yes" or "no" flag indicating whether or not the current allocation has a valid SMS Data Class associated with it and the share option for region was specified on the Data Class. Possible values are:

Y	Data Class information available
N	No associated Data Class information

See the associated variables DC_XREGION and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCSHXRF

DC_XSYSTEM

Format type = numeric
VSAM
SMS-Managed or Non-SMS-Managed

The DC_XSYSTEM variable contains the share option for system of the associated Data Class if and only if the related variable DC_XSYSTEM_FLAG = 'Y'. The DC_XSYSTEM_FLAG is 'Y' only when the data set has a valid SMS Data Class associated with it and the share option for system was specified on the Data Class.

See the associated variables DC_XSYSTEM_FLAG and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCSHXS

DC_XSYSTEM_FLAG

Format type = character
Maximum size = 1
Default = 'N'
VSAM
SMS-Managed or Non-SMS-Managed

The DC_XSYSTEM_FLAG variable is a "yes" or "no" flag indicating whether or not the current allocation has a valid SMS Data Class associated with it and the share option for system was specified on the Data Class. Possible values are:

Y	Data Class information available
N	No associated Data Class information

See the associated variables DC_XSYSTEM and DATACLAS. Also see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: DCSHXSF

DD

Format type = character
Maximum size = 8
Non-VSAM and VSAM

The DD variable contains the DD name associated with the current allocation request. For VSAM, this variable contains the DD name associated with the current allocation request for the Catalog Management base component.

DDD

Format type = character
Maximum size = 8
VSAM Data Component

The DDD variable contains the DD name associated with the current allocation request for the Catalog Management data component.

DDI

Format type = character
Maximum size = 8
VSAM Index Component

The DDI variable contains the DD name associated with the current allocation request for the Catalog Management index component.

DEF_DATACLAS

Format type = character
Maximum size = 8
Non-VSAM or VSAM
SMS-Managed or non-SMS-Managed

The DEF_DATACLAS variable contains the data class name associated with the resource owner of the data set (set only if RACF is installed and ACSDEFAULTS is YES in IGDSMSmm).

DEF_MGMTCLAS

Format type = character
Maximum size = 8
Non-VSAM or VSAM
SMS-Managed or non-SMS-Managed

The DEF_MGMTCLAS variable contains the management class name associated with the resource owner of the data set (set only if RACF is installed and ACSDEFAULTS is YES in IGDSMSmm).

DEF_STORCLAS

Format type = character
Maximum size = 8
Non-VSAM or VSAM
SMS-Managed or non-SMS-Managed

The DEF_STORCLAS variable contains the storage class name associated with the resource owner of the data set (set only if RACF is installed and ACSDEFAULTS is YES in IGDSMSmm).

DEFER_TAPE_MOUNT

Format type = character
Maximum size = 1
Non-VSAM

The DEFER_TAPE_MOUNT variable is a “Yes” or a “No” flag, indicating whether or not “DEFER” has been specified in the UNIT JCL parameter.

This attribute only has meaning for tape allocations. When specified, the system defers issuing the tape mount request (message IEF233A) until an open request is made for the data set. Possible values are:

Y	DEFER specified in the UNIT JCL parameter.
N	DEFER not specified in the UNIT JCL parameter.

This parameter is modifiable, allowing this parameter to be enabled or disabled as needed.

Alias: DEFER, DTM

DEVCLAS

Format type = character
Maximum size = 4
Non-VSAM

The DEVCLAS variable indicates the device class of the original unit of a new allocation. Possible values are:

DISK	The original unit was a disk device.
TAPE	The original unit was a tape device.

This input-only variable can be used to exclude all allocations of a particular device class from processing and is only applicable to new allocations intercepted in the ALLOC environment. For example, the following code in the ALLOC environment exempts tape allocations from processing:

```
IF &DEVCLAS = 'TAPE' THEN EXIT CODE(0)
```

DIRBLOCKS

Format type = numeric
Non-VSAM

The DIRBLOCKS variable contains the number of directory blocks requested for the allocation of a partitioned data set.

DISP

Format type = character
Maximum size = 3
Non-VSAM

The DISP variable represents the current disposition. The values it can contain are:

NEW MOD OLD SHR

DISPA

Format type = character
Maximum size = 7
Non-VSAM

The DISPA variable represents the abnormal disposition specified for this request. The values it can contain are:

KEEP DELETE CATLG UNCATLG

This variable can be changed only in the ALLOC environment.

DISPN

Format type = character
Maximum size = 7
Non-VSAM only

The DISPN variable represents the normal disposition specified for the request. The values it can contain are:

PASS KEEP DELETE CATLG UNCATLG

This variable can be changed only in the ALLOC environment.

Warning: Changing the DISPN value from PASS to another value can cause unpredictable results.

DMSGROUP

Format type = character
Maximum size = 8

For data set allocations requested by the BrightStor CA-Disk Auto-Restore STC, this variable contains the RACF group name for the user that initiated the Auto-Restore operation.

Note: Support in this area is currently limited to STCs named DMSAR that are either single-threaded or are processed by the BrightStor CA-Vantage Auto-Restore Manager.

DMSUSR **Format type = character**
Maximum size = 8

For data set allocations requested by the BrightStor CA-Disk Auto-Restore STC, this variable contains the USERID of the user that initiated the Auto-Restore operation.

Note: Support in this area is currently limited to STCs named "DMSAR" that are either single-threaded or are processed by version 3.1 of the BrightStor CA-Vantage Auto- Restore Manager.

DSN **Format type = character(sub)**
Maximum size = 44
Non-VSAM and VSAM

For non-VSAM, the DSN variable contains the data set name for the current allocation. For VSAM, the DSN variable contains the data set name from the Catalog Management base component control block.

For ACS, this value is established by the requestor of the processing ACS.

The DSN variable is a subscripted variable. Any node of the data set name can be accessed using the subscript operator. The maximum subscript is 22. Referring to a data set name node that does not exist returns the null string ''.

To get the second level qualifier in user variable C0, use the following statement:

```
SET &C0 = &DSN(2)
```

If no subscript is used, the entire data set name is used.

Note: For a DFHSM RECALL or RECOVER of a VSAM data set, this variable contains the name of the data set that is being recalled or recovered. For non-VSAM data sets, this variable contains the DFHSM "special" generated data set name. For more information, see "HSMDSN" and the section **DFSMShsm Support** in the chapter "**Concepts and Facilities**."

DSND

Format type = character(sub)
Maximum size = 44
VSAM Data Component

The DSND variable contains the data set name from the Catalog Management data component control block.

The DSND variable is a subscripted variable. Any node of the data set name may be accessed using the subscript operator. The maximum subscript is 22. Referring to a data set name node that does not exist returns the null string ' '. To get the second level qualifier in user variable C0, use the following statement:

```
SET &C0 = &DSND(2)
```

If no subscript is used, the entire data set name is used.

DSNI

Format type = character(sub)
Maximum size = 44
VSAM Index Component

The DSNI variable contains the data set name from the Catalog Management index component control block.

The DSNI variable is a subscripted variable. Any node of the data set name may be accessed using the subscript operator. The maximum subscript is 22. Referring to a data set name node that does not exist returns the null string ' '. To get the second level qualifier in user variable C0, use the following statement:

```
SET &C0 = &DSNI(2)
```

If no subscript is used, the entire data set name is used.

DSNTYPE

Format type = character
Maximum size = 7
Non-VSAM
SMS-Managed or non-SMS-Managed

The DSNTYPE variable contains the data set name type as follows:

LIBRARY	for a PDSE
PDS	Allocate a standard PDS
EXC	Extended conditional. If the needed system resources are available, allocate a PDSE. Otherwise, allocate a PDS.
EXR	Extended required. If the needed system resources are available, allocate a PDSE. Otherwise, fail the allocation.
HFS	Hierarchical file system PIPE---FIFO special file
null	for all other types of data sets.

For ACS, this value is established by the requestor of the processing ACS.

DSORG

Format type = character
Maximum size = 3
Non-VSAM and VSAM

The DSORG variable contains the data set organization that was either specified or implied in the JCL for the current allocation. For example, if 'DSORG=DA' is coded in the JCL, then the DSORG variable contains 'DA'. If no DSORG is coded, DSORG contains the implied value 'PS' unless a PDS is implied by directory blocks in the SPACE parameter, in which case the DSORG variable contains 'PO'.

This variable can be changed in the ALLOC and SPACE environments to any valid JCL specification, with three exceptions. The DSORG variable can neither be changed to nor changed from 'IS', 'ISU', or 'VS'.

For ACS, this value is established by the requestor of the processing ACS.

Note: For ACS processing, the DSORG variable is only used as input and cannot be changed. If no DSORG is coded in the JCL, the variable is not available. When the 'LIKE' parameter is used on the JCL DD statement, the DSORG value is not available from the 'LIKE' data set as input to the ACS routines (refer to the MVS/DFP Storage Administration Reference (SC26-4566) Chapter 15, 'Constraints on READ-ONLY Variable Usage' or IBM Apar OY45066).

Note: While a ‘DSORG=PO’ parameter is not required in the JCL to allocate a partitioned data set, it is always required that directory blocks be specified. This means that when DIRBLOCKS is non-zero, it is always a partitioned data set that is being allocated, even when the DSORG variable value is missing.

DSOWNER

Format type = character
Maximum size = 8
Non-VSAM and VSAM
SMS-Managed or non-SMS-Managed

The DSOWNER variable contains the name of the user or group that owns the data set. This variable is valid only if RACF is installed.

For ACS, this value is established by the requestor of the processing ACS.

DSTYPE

Format type = character
Maximum size = 4
Non-VSAM and VSAM

The DSTYPE variable represents the type of data set currently being processed. It is primarily determined by interrogating the indicator bytes in the JFCB. Possible values are:

GDS	Generation Data set
TEMP	Temporary Data set
PERM	Permanent Data set.

For ACS, this value is established by the requestor of the processing ACS.

Note: The operating system considers data sets with Step Termination Dispositions (&DISPN variable) of 'PASS' or 'DELETE' to be 'TEMP'.

Note: While processing in the QSCAN and QUOTA environments, this variable is only available for input during a DFHSM RECALL or RECOVER of a non-VSAM data set.

EXPDT

Format type = numeric
Non-VSAM and VSAM

The EXPDT variable contains the user-requested expiration date and can be reset to a new value. The expiration date value is in Julian notation (ccyyddd).

The following statement causes a data set to expire on the last day of 1999:

```
SET &EXPDT = 1999365
```

This statement causes no expiration date to be enforced.

```
SET &EXPDT=0
```

See also the description of variable "RETPD." Note that use of both EXPDT and RETPD in an ASR is not recommended. If both EXPDT and RETPD are assigned a new value in the ASR, the value assigned to EXPDT takes precedence over that assigned to RETPD.

For ACS, this value is established by the requestor of the processing ACS.

Note: For SMS-managed allocations, BrightStor CA-Allocate's ability to modify this variable is contingent upon having access to the appropriate system control blocks.

EXPDT_CODED

Format type = character
Maximum size = 1
Default = 'N'
Non-VSAM

The EXPDT_CODED variable is a "yes" or "no" flag indicating whether or not the expiration date was coded in JCL. This variable is applicable only in the ALLOC and OLD environments. It has a blank value in all other environments. Possible values are:

Y	Indicates the expiration date was coded in JCL
N	Indicates that it was not coded

EXTENDCOMP

Format type = character
VSAM
SMS-Managed or non-SMS-Managed

The EXTENDCOMP variable helps eliminate space abends for VSAM clusters. It allows you to control the allocation attributes of the next extent for either the DATA or INDEX component. If used properly, this variable can prevent online outages caused by space abends. Possible values are:

DATA Represents the DATA component of the cluster currently being extended

INDEX Represents the INDEX component of the cluster currently being extended

For related information, see the variable description for "EXTENDVOL."

For general information on VSAM support within the EXTEND environment, see the section **VSAM Processing in EXTEND** in the chapter "**Concepts and Facilities**."

For an example of how to code this variable, see **FIGURE 2-5 Sample EXTEND Environment** ASR for VSAM.

Alias: EC

EXTENDVOL

Format type = character
VSAM
SMS-Managed or non-SMS-Managed

The EXTENDVOL variable represents the type of volume the new extent is allocated on. Depending on the value set for this variable, the next extent can either be allocated onto the current volume or, after successfully recovering from an End-Of-Volume condition, onto a new volume. Possible values are:

CURRENT	Identifies the current volume as the volume being extended on
NEW	Forces the next extent onto a NEW volume.

For related information, see the variable description for "EXTENDCOMP."

For general information on VSAM support within the EXTEND environment, see the section **VSAM Processing in EXTEND** in the chapter "**Concepts and Facilities**."

For an example of how to code this variable, see **FIGURE 2-5 Sample EXTEND Environment** ASR for VSAM.

Alias: EV

EXTENTS

Format type = numeric
Non-VSAM and VSAM

The EXTENTS variable contains a number from 0 to 123, indicating how many extents the data set is in on the current volume.

EXTENTSD

Format type = numeric
VSAM

The EXTENTSD variable contains a value from 0 to 123, indicating the total number of extents currently allocated to the data component of the VSAM data set. The ASR can determine whether the data component caused the EOV_VSAM environment to be entered by checking ABENDCOMP for a value of DATA.

EXTENTSI

Format type = numeric
VSAM

The EXTENTSI variable contains a value from 0 to 123, indicating the total number of extents currently allocated to the index component of the VSAM data set. The ASR can determine whether the index component caused the EOV_VSAM environment to be entered by checking ABENDCOMP for a value of INDEX.

FAILIFNOVOLSEL

Format type = character
Maximum size = 1
Default = 'N'
Non-VSAM and VSAM

When volume selection is requested by setting a Storage Group in the ASR, BrightStor CA-Allocate attempts to do volume selection. If it is unable to do so, message VAM0039 is issued describing the reason(s) a volume was not selected. The FAILIFNOVOLSEL, abbreviated FINVS, determines whether BrightStor CA-Allocate fails the allocation or lets IBM try to complete the allocation, using the original volume(s) specified.

Alias: FINVS

FAILMOD

Format type = character
Maximum size = 2
Default = 'VA'
VSAM

The FAILMOD variable sets the module suffix in VSAM allocation failure error message IDC3009I to show that BrightStor CA-Allocate is the failing component when the ASR returns a non zero return code via EXIT CODE(n).

Variable FAILMOD specifies the 'aa' portion of the message below the module returning the error. The default value is 'VA'. To isolate the failing area in BrightStor CA-Allocate, the ASR can set different values in different areas.

```
IDC3009I VSAM CATALOG RETURN CODE IS rc-REASON CODE IS IGGOCLaa-crs
```

FDR

Format type = character
Maximum size = 1
Default = 'N'
Non-VSAM
Non-SMS-Managed

The FDR variable is a flag indicating whether or not the current allocation is being intercepted by the FDR API. Possible values are:

Y	Identifies the allocation to the ALLOC environment as one that has been intercepted by the FDR API.
N	(default) Identifies the allocation to the ALLOC environment as one that ha not been intercepted by the FDR API.

For general information on the FDR API, see the section **FDR Support** in the chapter "**Concepts and Facilities**."

For an example of how to code this variable, see **FIGURE 2-21 Sample ASR that** incorporates the FDR Support.

FILENUM

Format type = numeric
Maximum size = 4
Non-VSAM

The FILENUM variable represents the data-set-sequence-number specified in the LABEL parameter. This number is used to identify the relative position of a data set on a tape volume. The values it can contain range from 1 through 9999. It is applicable to new tape data sets only.

This variable can be changed only in the ALLOC environment and only to a positive integer. If FILENUM is set to zero, the system defaults it to 1. Careful consideration should be taken before modifying this variable. For example, arbitrarily changing the FILENUM variable from 1 to 3 for a nonspecific tape allocation causes the allocation to fail with an '**IEC145I 413-34**' message.

For TAPE-to-DISK device-type conversions, the FILENUM variable is automatically reset to 1. For DISK-to-TAPE device-type conversions, unless a new value is requested, whatever was specified for the original disk allocation (typically 1) is used. A detailed explanation of why different actions are taken, depending on the conversion type, can be found in the section **Device-Type Conversions** in the chapter "**Concepts and Facilities**."

FIXUNITMISMATCH

Format type = character
Maximum size = 8
Non-VSAM
Non-SMS-Managed

The FIXUNITMISMATCH variable contains the device type associated with the catalog entry of a data set. For disk data sets, this is the esoteric name 'SYSALLDA', and for tapes, the associated generic name (that is, '3480', '3420', and so on).

This information can be used by the ASR to correct invalid UNIT information in order to allow the successful allocation of an old data set. For more information about how and when this variable is used, see the section **OLD Environment** in the chapter "**Concepts and Facilities**" and the description of variable "UNITMISMATCH."

Alias: FUMM

GROUP

Format type = character
Maximum size = 8
Non-VSAM and VSAM

The GROUP variable contains the name of the group to which the user is currently connected. The GROUP variable is retrieved only if RACF is the security system. If another security package is used, you can write a User Exit to associate a user with a group or use the FILTLIST statement to build a list of users.

For ACS, this value is established by the requestor of the processing ACS.

Note: For a DFHSM RECALL or RECOVER of a VSAM data set, this variable contains the RACF group name to which the issuer of the allocation request is currently connected. This should always be the DFHSM address space's current connect group. See the description of variable "HSMDSN" and the section **DFSMShsm Support** in the chapter "**Concepts and Facilities**" for more information.

GUARANTEED_SPACE

Format type = character
Maximum size = 1
SMS-Managed

This variable indicates whether the current Storage Class (for example, the SC variable) has the "Guaranteed Space" attribute associated with it. Possible values are:

Y	The "Guaranteed Space" attribute is associated with the current Storage Class
N	The "Guaranteed Space" attribute is NOT associated with the current Storage Class
?	BrightStor CA-Allocate is unable to determine if the "Guaranteed Space" attribute is associated with the current Storage Class.

Alias: GSA

HLQ

Format type = character
Maximum size = 8
Non-VSAM and VSAM

For non-VSAM, the HLQ variable contains the high-level qualifier of the data set name.

For VSAM, the HLQ variable contains the high-level qualifier of the cluster.

For ACS, this value is established by the requestor of the processing ACS.

For both non-VSAM and VSAM, HLQ contains the same value as &DSN(1).

Note: For a DFHSM RECALL or RECOVER of a non-VSAM data set, this variable contains the high-level qualifier of the DFHSM "special" generated data set name. See the description of variable "HSMHLQ" and the section **DFSMShsm Support** in the chapter "**Concepts and Facilities**" for more information.

HLQD

Format type = character
Maximum size = 8
VSAM Data Component

The HLQD variable contains the high-level qualifier of the data component name. This is the same value as &DSND(1).

HLQI

Format type = character
Maximum size = 8
VSAM Index Component

The HLQI variable contains the high-level qualifier of the index component name. This is the same value as &DSNI(1).

HSMDSN

Format type = character(sub)
Maximum size = 44

The HSMDSN variable contains the "requested" data set name that results from a DFSMShsm RECALL or RECOVER operation of a non-VSAM data set. See the section **DFSMShsm Support** in the chapter "**Concepts and Facilities**" for more information.

The HSMDSN variable is a subscripted variable. Any node of the data set name can be accessed using the subscript operator. The maximum subscript is 22. Referring to a data set name node that does not exist returns the null (`') string.

For example, to get the second level qualifier in user variable C0, use the following statement:

```
SET &C0 = &HSMDSN(2)
```

If no subscript is used, the entire data set name is returned.

Note: While processing in the ALLOC and SPACE environments, this variable is only available for input during a DFHSM RECALL or RECOVER of a non-VSAM data set.

Note: While processing in the QUOTA environment, this variable is only available for input when 'QUOTAFUNC=ALLOC' during a DFHSM RECALL or RECOVER.

HSMFUNC

Format type = character
Maximum size = 8

The HSMFUNC variable contains the name of the DFSMShsm function or blank. Valid values are:

Value	DFSMShsm Data Set Operation
RECALL	DFSMShsm Recall
RECOVER	DFSMShsm Recover
<blank>	Not one of the above DFSMShsm operations

HSMGROUP

Format type = character
Maximum size = 8

The HSMGROUP variable contains the RACF group name for the user initiating the DFSMShsm RECALL or RECOVER operation. See the section **DFSMShsm Support** in the chapter "**Concepts and Facilities**" for more information.

The HSMGROUP variable has a value only if RACF is the security system. If another security package is used, it is possible to write a User Exit to associate a user with a group. Or, the FILTLIST statement can be used to build a list or group of users.

Note: While processing in the ALLOC and SPACE environments, this variable is only available for input during a DFHSM RECALL or RECOVER of a non-VSAM data set.

Note: While processing in the QUOTA environment, this variable is only available for input when 'QUOTAFUNC=ALLOC' during a DFHSM RECALL or RECOVER.

HSMHLQ

Format type = character
Maximum size = 8

The HSMHLQ variable contains the high-level qualifier of the requested non-VSAM data set name that result from a DFSMShsm RECALL or RECOVER operation. See the section **DFSMShsm Support** in the chapter "**Concepts and Facilities**" for more information.

Note: While processing in the ALLOC and SPACE environments, this variable is only available for input during a DFHSM RECALL or RECOVER of a non-VSAM data set.

Note: While processing in the QUOTA environment, this variable is only available for input when 'QUOTAFUNC=ALLOC' during a DFHSM RECALL or RECOVER.

HSMJOB

Format type = character
Maximum size = 8

The HSMJOB variable contains the job name or TSO user ID that requested the non-VSAM data set that results from a DFSMShsm RECALL or RECOVER operation. See the section **DFSMShsm Support** in the chapter "**Concepts and Facilities**" for more information.

Note: While processing in the ALLOC and SPACE environments, this variable is only available for input during a DFHSM RECALL or RECOVER of a non-VSAM data set.

Note: While processing in the QUOTA environment, this variable is only available for input when 'QUOTAFUNC=ALLOC' during a DFHSM RECALL or RECOVER.

HSMLLQ

Format type = character
Maximum size = 8

The HSMLLQ variable contains the low-level qualifier of the requested non-VSAM data set name that results from a DFSMShsm RECALL or RECOVER operation. See the section **DFSMShsm Support** in the chapter "**Concepts and Facilities**" for more information.

Note: While processing in the ALLOC and SPACE environments, this variable is only available for input during a DFHSM RECALL or RECOVER of a non-VSAM data set.

Note: While processing in the QUOTA environment, this variable is only available for input when 'QUOTAFUNC=ALLOC' during a DFHSM RECALL or RECOVER.

HSMNQUAL

Format type = numeric

The HSMNQUAL variable contains the number of qualifiers (nodes) in the requested non-VSAM data set name that results from a DFSMShsm RECALL or RECOVER operation. See the section **DFSMShsm Support** in the chapter "**Concepts and Facilities**" for more information.

Note: While processing in the ALLOC and SPACE environments, this variable is only available for input during a DFHSM RECALL or RECOVER of a non-VSAM data set.

Note: While processing in the QUOTA environment, this variable is only available for input when 'QUOTAFUNC=ALLOC' during a DFHSM RECALL or RECOVER.

HSMUSER **Format type = character**
Maximum size = 8

The HSMUSER variable contains the RACF user name for the user initiating the DFSMShsm RECALL or RECOVER operation. This variable is retrieved from the security system if RACF, eTrust CA-TOP SECRET, or CA-ACF2 is active. Otherwise, it is the TSO user ID in TSO, or null (`') in batch. See the section **DFSMShsm Support** in the chapter "**Concepts and Facilities**" for more information.

Note: While processing in the ALLOC and SPACE environments, this variable is only available for input during a DFHSM RECALL or RECOVER of a non-VSAM data set.

Note: While processing in the QUOTA environment, this variable is only available for input when 'QUOTAFUNC=ALLOC' during a DFHSM RECALL or RECOVER.

IFALREADYCAT

Format type = character
Maximum size = 1

This variable is applicable only at sites running DFP Release 3 and later.

The IFALREADYCAT, abbreviated IFCAT, variable contains the catalog status of the data set name being allocated. If the data set name is already cataloged, IFALREADYCAT contains 'Y'. If the data set name is not cataloged, IFALREADYCAT contains 'N'. Use IFALREADYCAT to determine when data sets with duplicate names are being allocated.

Value	Description
Y	Data set name already cataloged
N	Data set name is not already cataloged
?	Unable to determine whether there is catalog entry for the data set name.

For information about enqueues and SMS-managed GDGs, see the section **SMS-Managed GDGs** in the chapter "**Introduction**."

Note: See also the description of variable IFALREADYCATVOL, it holds the volume serial number of the volume where the data set is cataloged.

Note: See STOP_NOT_CATLG2 for information about possible methods to deal with an IFALREADYCAT = 'Y' situation.

Alias: IFCAT

IFALREADYCATVOL

Format type = character
Maximum size = 6

This variable is applicable only at sites running DFP Release 3 and above.

Use this variable with IFALREADYCAT (indicates if the data set name being allocated is already cataloged). When IFALREADYCAT contains a 'Y', the IFALREADYCATVOL variable contains the volume serial number of the volume where the data set is already cataloged. If the data set name being allocated is not cataloged, IFALREADYCATVOL contains spaces.

Alias: IFCATV

IMBED

Format type = character
Maximum size = 1
VSAM

The IMBED variable is a “yes” or “no” flag indicating whether or not the IMBED attribute was detected while defining a VSAM KSDS data set. Possible values are:

Y	Attribute was specified.
N	Attribute was not specified.

This variable can be changed only in the DEFINE environment. Possible values are:

Y	Add this attribute.
N	Remove this attribute

INDEXSEP

Format type = character
Maximum size = 1
VSAM
Non-SMS-Managed only

The INDEXSEP variable controls whether the VSAM Index Component should be placed on the same volume as the associated VSAM data component (INDEXSEP=N) or on a different volume (INDEXSEP=Y).

Note: If this variable is not set, the index is separated from the data only if the originally requested index volumes are different from the data volumes.

INDEXSUFFIX

Format type = character
Maximum size = 8
VSAM Index Component

The INDEXSUFFIX variable is used to generate the index component name based on the base cluster name. The value of variable INDEXSUFFIX is appended to end of the base cluster name to create the index component name. This is typically used to make the index component name end in ".INDEX" or another installation standard.

The initial value of INDEXSUFFIX is a null string. If logic in the ASR sets INDEXSUFFIX to a non-null value, then the index component name is generated or replaced when the "EXIT CODE(0)" ASR statement is executed. Note that the value of the variable DSNI is not altered when INDEXSUFFIX is assigned a value.

For example, if the base cluster name is MY.VSAM.FILE, the ASR statement below results in the index component name being set to MY.VSAM.FILE.INDEX when the "EXIT CODE(0)" ASR statement is executed.

```
SET &INDEXSUFFIX = 'INDEX'
```

JOB

Format type = character
Maximum size = 8
Non-VSAM and VSAM

The JOB variable contains the job name requesting the allocation, if the execution mode is BATCH or TASK.

For a DFHSM RECALL or RECOVER operation, this variable contains the job name of the issuer allocation request. This should always be the jobname of the DFHSM address space. See the description of variable "HSMJOB" and also the section **DFSMShsm Support** in the chapter "**Concepts and Facilities**" for more information.

JOBCATOK

Format type = character
Maximum size = 1
Default = 'Y'
VSAM

The JOBCATOK variable determines whether to use (JOBCATOK=Y) or disallow (JOBCATOK=N) the catalog allocated through JOBCAT. If the job being processed does not have a JOBCAT DD statement allocated to it, changing this variable has no effect on the DEFINE.

This variable can be used in conjunction with variables CATONDEFOK and STEPCATOK to enforce installation standards for catalog use, as shown in this example:

```
SET &CATONDEFOK = 'N'
SET &STEPCATOK = 'N'
SET &JOBCATOK = 'N'
WRITE 'The proper catalog to use is &CATDSN'
```

If the JOBCATOK variable changes, the values of variables CATDSN, CATNQUAL, CATHLQ, and CATLLQ can also change. Also see the descriptions of the CATDSN, CATNQUAL, CATONDEFOK, STEPCATOK, CATHLQ, and CATLLQ variables.

JOBCLASS

Format type = character
Maximum size = 1
Non-VSAM and VSAM

The JOBCLASS variable contains the job class associated with the requesting allocation.

Alias: JC

LABEL

Format type = character
Maximum size = 3
Non-VSAM

The LABEL variable contains the label type specified in the LABEL parameter and applies to new tape data sets only. Possible values are:

SL	IBM standard label
SUL	both IBM standard and user labels
AL	ISO / ANSI V1 or ISO / ANSI / FIPS V3 labels
AUL	both AL and user labels
NSL	nonstandard label
NL	no label
BLP	the system is to bypass label processing
LTM	the data set has a leading tapemark

This variable can be changed only in the ALLOC environment and only to one of the values listed above. Exercise caution when changing this variable. Do not set LABEL to a value your installation does not support.

For TAPE-to-DISK device-type conversions, the LABEL variable is automatically reset to SL. For DISK-to-TAPE device-type conversions, unless a new value is requested, the value specified or implied for the original disk allocation (typically SL) is used. A detailed explanation of why different actions are taken, depending on the conversion type, can be found the section **Device-Type Conversions** in the chapter ”**Concepts and Facilities**.”

LARGEST_EXTCYL

Format type = numeric
Non-VSAM and VSAM

This LSPACE support variable contains the largest number of contiguous cylinders available for allocation on the volume or the storage group. The value varies, depending on how other variables are specified.

For details, see the section **LSPACE Support for Primary and** Secondary Allocation in the chapter “**Concepts and Facilities**.”

Alias: LEC

LARGEST_EXTCYL_NOT

Format type = numeric
Non-VSAM and VSAM

This LSPACE support variable contains the largest number of cylinders available for allocation in the storage group without violating the freespace threshold. The value varies, depending on how other variables are specified.

For details, see the section **LSPACE Support for Primary and** Secondary Allocation in the chapter "**Concepts and Facilities**."

Alias: LECN

LARGEST_EXTMB

Format type = numeric
Non-VSAM and VSAM

This LSPACE support variable contains the largest number of contiguous megabytes available for allocation on the volume or in the entire storage group. The value varies, depending on how other variables are specified.

For details, see the section **LSPACE Support for Primary and** Secondary Allocation in the chapter "**Concepts and Facilities**."

Alias: LEMB

LARGEST_EXTMB_NOT

Format type = numeric
Non-VSAM and VSAM

This LSPACE support variable contains the largest megabyte data area available for allocation in the storage group without violating the freespace threshold. The value varies, depending on how other variables are specified.

For details, see the section **LSPACE Support for Primary and** Secondary Allocation in the chapter "**Concepts and Facilities**."

Alias: LEMBN

LARGEST_EXTTRK

Format type = numeric
Non-VSAM and VSAM

This LSPACE support variable contains the largest number of contiguous tracks available for allocation on the volume or in the entire storage group. The value varies, depending on how other variables are specified.

For details, see the section **LSPACE Support for Primary and** Secondary Allocation in the chapter "**Concepts and Facilities**."

Alias: LET

LARGEST_EXTTRK_NOT

Format type = numeric
Non-VSAM and VSAM

This LSPACE support variable contains the largest number of tracks available for allocation in the storage group without violating the freespace threshold. The value varies, depending on how other variables are specified.

For details, see the section **LSPACE Support for Primary and** Secondary Allocation in the chapter "**Concepts and Facilities**."

Alias: LETN

LIKE

Format type = character
Maximum size = 44
Non-VSAM

This variable applies only if SMS is turned on for the system. This variable contains the name of the LIKE=data.set.name if the DD LIKE parameter was coded in JCL. If it was coded, you can either update the variable with the DSN to use, or you can remove the LIKE parameter. If it was not coded, you can add the LIKE parameter.

LIKEHLQ

Format type = character
Maximum size = 8
Non-VSAM

This variable contains the High Level Qualifier of the LIKE DSN if it was coded. It applies only if SMS is turned on for the system.

LIKELLQ

Format type = character
Maximum size = 8
Non-VSAM

This variable contains the Low Level Qualifier of the LIKE DSN if it was coded. It applies only if SMS is turned on for the system.

LIKENQUAL

Format type = numeric
Non-VSAM

This variable contains the number of qualifiers in the LIKE DSN if it is coded. It applies only if SMS is turned on for the system.

LLQ

Format type = character
Maximum size = 8
Non-VSAM and VSAM

For non-VSAM, the LLQ variable contains the low-level qualifier of the data set name.

For VSAM, the LLQ variable contains the low-level qualifier of the base cluster name.

For ACS, this value is established by the requestor of the processing ACS.

For both non-VSAM and VSAM, LLQ contains the same value as &DSN(&NQUAL).

Note: For a DFHSM RECALL or RECOVER of a non-VSAM data set, this variable contains the low-level qualifier of the DFHSM "special" generated data set name. See the description of variable "HSMLLQ" and the section **DFSMShsm Support** in the chapter "**Concepts and Facilities**" for more information.

LLQD

Format type = character
Maximum size = 8
VSAM Data Component

The LLQD variable contains the low-level qualifier of the data component name. This is the same value as &DSND(&NQUALD).

LLQI

Format type = character
Maximum size = 8
VSAM Index Component

The LLQI variable contains the low-level qualifier of the index component name. This is the same value as &DSNI(&NQUALI).

LRECL

Format type = numeric
Non-VSAM and VSAM

For non-VSAM, the LRECL variable contains the logical record length specified for the data set.

For VSAM, the LRECL variable contains the average record size of the data component.

Note: This variable is not currently available for VSAM in the EXTEND environment.

LSPACE_FREESPACE

Format type = character
Maximum size = 1
Non-VSAM and VSAM

This LSPACE support variable indicates whether the free-space threshold was violated. It only applies if a storage group (not a volume) is being processed. Possible values are:

N This is the default value and indicates that the free-space threshold was not violated.

Y The free-space threshold was violated for LSPACE processing.

For details, see the section **LSPACE Support for Primary and** Secondary Allocation in the chapter "**Concepts and Facilities**."

Alias: LFF

LSPACE_PCTAFT_CYL

Format type = numeric
Non-VSAM and VSAM

If the LARGEST_EXTCYL was used during allocation, this LSPACE support variable contains the updated free-space percent (rounded to the nearest whole number) for the volume.

For details, see the section **LSPACE Support for Primary and** Secondary Allocation in the chapter "**Concepts and Facilities**."

Alias: LPAC

LSPACE_PCTAFT_CYLNOT

Format type = numeric
Non-VSAM and VSAM

If the LARGEST_EXTCYL_NOT was used during allocation, this LSPACE support variable contains the updated free-space percent (rounded to the nearest whole number) for the volume.

For details, see the section **LSPACE Support for Primary and** Secondary Allocation in the chapter "**Concepts and Facilities**."

Alias: LPACN

LSPACE_PCTAFT_MB

Format type = numeric
Non-VSAM and VSAM

If the LARGEST_EXTMB was used during allocation, this LSPACE support variable contains the updated free-space percent (rounded to the nearest whole number) for the volume.

For details, see the section **LSPACE Support for Primary and** Secondary Allocation in the chapter "**Concepts and Facilities**."

Alias: LPAM

LSPACE_PCTAFT_MBNOT

Format type = numeric
Non-VSAM and VSAM

If the LARGEST_EXTMB_NOT was used during allocation, this LSPACE support variable contains the updated free-space percent (rounded to the nearest whole number) for the volume.

For details, see the section **LSPACE Support for Primary and** Secondary Allocation in the chapter "**Concepts and Facilities**."

Alias: LPAMN

LSPACE_PCTAFT_TRK

Format type = numeric
Non-VSAM and VSAM

If the LARGEST_EXTTRK was used during allocation, this LSPACE support variable contains the updated free-space percent (rounded to the nearest whole number) for the volume.

For details, see the section **LSPACE Support for Primary and** Secondary Allocation in the chapter "**Concepts and Facilities**."

Alias: LPAT

LSPACE_PCTAFT_TRKNOT

Format type = numeric
Non-VSAM and VSAM

If the LARGEST_EXTTRK_NOT was used during allocation, this LSPACE support variable contains the updated free-space percent (rounded to the nearest whole number) for the volume.

For details, see the section **LSPACE Support for Primary and** Secondary Allocation in the chapter "**Concepts and Facilities**."

Alias: LPATN

LSPACE_RETURN_CODE

Format type = numeric
Non-VSAM and VSAM

This LSPACE support variable indicates whether to consider the LSPACE output variables valid or not.

Possible values are:

ZERO	Valid values were returned to all output variables.
NON-ZERO	One or more output variables are invalid.

Alias: LRC

Below is the list of NON-ZERO return codes and their meanings:

TABLE 4-4 Possible Values for the LRC Variable

RC	Meaning
1	The requested volume name is invalid or the volume is offline.
2	LSPACE failed. Unable to return extent information. See VAM7000 message description.
3	STORGRP Name is invalid or there are zero volumes available.
4	One or more of the LSPACE allocation amount variables is 0.
5	BrightStor CA-Allocate failed to get the number of UCBs.
6	No UCBs in the unit or esoteric group.
7	BrightStor CA-Allocate failed to get the list of UCBs.
8	Invalid Function Call. Must be LV or LS.
9	BrightStor CA-Allocate failed to get the length of STORGRP.
10	BrightStor CA-Allocate failed to get storage to hold the STORGRP.
11	BrightStor CA-Allocate failed to get the volumes in STORGRP.
12	No volumes in unit match STORGRP.
13	Unable to obtain the volumes in a SMS Storage Group.
16	An attempt was made to invoke LSPACE support from an environment in which it is not supported.

LSPACE_STORGRP

Format type = character
Maximum size = 134
Non-VSAM and VSAM
SMS-Managed or non-SMS-Managed

This LSPACE support variable specifies the storage group on which to do LSPACE.

Alias: LS

LSPACE_VOLUME

Format type = character
Maximum size = 6
Non-VSAM and VSAM
SMS-Managed or non-SMS-Managed

This LSPACE support variable specifies the DASD volume on which to do LSPACE. In the EXTEND environment, this value defaults to the volume on which the allocation is being done.

Alias: LV

MAXSIZE

Format type = numeric
Non-VSAM

The MAXSIZE variable contains the maximum amount of space a data set can ideally acquire on the first volume. The value is specified in tracks, cylinders, or blocks depending on the value contained in the SPACTYPE variable.

MAXSIZE is computed by multiplying the secondary allocation quantity by 15 and adding the primary space quantity. Fragmentation or lack of free space can reduce the space the data set can actually acquire.

MGMTCLAS

Format type = character
Maximum size = 8
Non-VSAM and VSAM
SMS-Managed only

The MGMTCLAS variable contains the SMS management class construct name. The original value in the MGMTCLAS variable comes from the original user request or any IBM ACS processing. In the ACS environment, the value can be set to any valid management class name or removed by setting it to null.

If the STORCLAS has not been set, the data set is not SMS-managed, so BrightStor CA-Allocate ignores any attempt to set MGMTCLAS, because it is not a valid variable.

The Management Class name may not always be available in the DEFINE, SCRATCH, and SPACE environments or in the QUOTA environment when QUOTAFUNC=SCRATCH.

Alias: MANAGEMENTCLASS or MC

MODULE

Format type = character
Maximum size = 8
Non-VSAM and VSAM

The MODULE variable contains the name of the control section that caused the allocation to occur.

MSPOLICY

Format type = character
Maximum size = 8

The MSPOLICY variable identifies the name of a management policy associated with tape data for a tape management system-driven allocation. This variable is data for a tape management system-driven allocation. This variable is set by the DFSMSrmm EDGUX100 installation exit or by the IBM pre-ACS installation exit.

MSPOOL

Format type = character
Maximum size = 8

The MSPOOL variable identifies the tape pool name associated with the data set being allocated. This variable is set by the DFSMSrmm EDGUX100 installation exit or by the IBM pre-ACS installation exit.

MSVGP

Format type = character
Maximum size = 8
Non-VSAM

The MSVGP variable contains the name of the mass storage volume group requested by the user.

For ACS, this value is established by the requestor of the processing ACS.

MVOLSPC

Format type = character
Maximum size = 1
Default = 'D'
Non-VSAM and VSAM

The MVOLSPC variable specifies the action to be taken when sufficient volumes cannot be found to satisfy a multi-volume data set allocation. It is applicable only in the ALLOC and DEFINE environments.

This condition can occur because:

1. Sufficient volumes are not available due to lack of free-space.
2. The storage group does not contain enough volumes to satisfy the request.

Set MVOLSPC to a single character (D, A, or F) to select the appropriate action to be taken:

D Decrement the number of volumes requested to the number of volumes available. As long as at least one volume can be found that can hold the PRIMARY allocation, the allocation is considered successful.

A Allow additional volumes, even if free-space requested is not currently available.

F BrightStor CA-Allocate does not modify the allocation and lets MVS attempt to allocate it as it was originally requested. If &FINVS = 'Y', BrightStor CA-Allocate fails the allocation.

NEWNAME

Format type = character(sub)
Maximum size = 44
Non-VSAM and VSAM

The NEWNAME variable has meaning only during a data set rename operation. It is the new name given to the data set. It is available only in the RENAME and QUOTA (when &QUOTAFUNC='RENAME') environments.

NEWNAMEHLQ

Format type = character
Maximum size = 8
Non-VSAM and VSAM

The NEWNAMEHLQ variable contains the high-level qualifier of the new data set name when the data set is being renamed. This is the same value as &NEWNAME(1). It is available only in the RENAME and QUOTA (when &QUOTAFUNC='RENAME') environments.

NEWNAMELLQ

Format type = character
Maximum size = 8
Non-VSAM and VSAM

The NEWNAMELLQ variable contains the low-level qualifier of the new data set name when the data set is being renamed. This is the same value as &NEWNAME(&NEWNAMENQUAL). It is available only in the RENAME and QUOTA (when &QUOTAFUNC='RENAME') environments.

NEWNAMENQUAL

Format type = numeric
Non-VSAM and VSAM

The NEWNAMENQUAL variable is equal to the number of qualifiers (nodes) in the new data set name during a data set rename. For example, when &NEWNAME=MY.DATA.SET, &NEWNAMENQUAL is 3. It is available only in the RENAME and QUOTA (when &QUOTAFUNC='RENAME') environments.

NKEYRANGE

Format type = numeric
VSAM

The NKEYRANGE variable is equal to the number of key ranges contained in the data component.

NQUAL

Format type = numeric
Non-VSAM and VSAM

For non-VSAM, the NQUAL variable is equal to the number of qualifiers (nodes) in the data set name.

For ACS, this value is established by the requestor of the processing ACS.

For VSAM, the NQUAL variable is equal to the number of qualifiers (nodes) in the base cluster name.

For example, the data set name MY.VSAM.FILE sets NQUAL to 3.

Note: For a DFHSM RECALL or RECOVER of a non-VSAM data set, this variable contains the number of qualifiers (nodes) in the DFHSM "special" generated data set name. See "HSMNQUAL" and the section **DFSMShsm Support** in the chapter "**Concepts and Facilities**" for more information.

NQUALD

Format type = numeric
VSAM Data Component

The NQUALD variable is equal to the number of qualifiers (nodes) in the data component name. For example, the data component name 'MY.VSAM.FILE.DATA' sets NQUALD to 4.

NQUALI

Format type = numeric
VSAM Index Component

The NQUALI variable is equal to the number of qualifiers (nodes) in the index component name. For example, the index component name 'MY.VSAM.FILE.INDEX' sets NQUALI to 4.

NTRKD

Format type = numeric
VSAM

The NTRKD variable contains the total number of tracks currently allocated to the data component of the VSAM data set. By seeing if the ABENDCOMP variable is equal to DATA you can determine whether the data component caused the EOV_VSAM environment to be entered.

NTRKI

Format type = numeric
VSAM

The NTRKI variable contains the total number of tracks currently allocated to the index component of the VSAM data set. By seeing if the ABENDCOMP variable equals INDEX, you can determine whether the index component caused the EOV_VSAM environment to be entered.

NUNIT

Format type = numeric
Non-VSAM and VSAM

The NUNIT variable initially contains the number of units specified for the allocation. For JCL allocations, if UNIT=(3380,5) is specified, NUNIT contains a value of 5. If the JCL unit count field is not specified with the unit parameter, the NUNIT value is 1.

For VSAM, NUNIT is obtained from the Catalog Management base component control block.

For non-VSAM, NUNIT is a powerful output variable. It can be set to a number greater than one to allow an allocation to become multi-volume. Setting NUNIT to greater than 1 in the ALLOC environment is one way to avoid x37 abends for sequential data sets, although a better way to avoid x37 abends is to set &POOLSUB = 'Y' in the EOV environment.

Any value specified for NUNIT is reduced as needed so that it does not exceed the number of units eligible as defined by the UNIT parameter or other ASR specifications. When NUNIT is greater than one, any resulting allocation is initially passed a single volume. If writing to the data set causes MVS to attempt to extend to additional volumes, MVS is able to pick another eligible volume for the data set. Note that BrightStor CA-Allocate does not take part in selecting additional volumes for the data set. MVS chooses the volumes based on normal DADSM allocation rules.

To allow multi-volume allocations when they are needed while not allocating single volume data sets to multiple volumes, set NVOL=1, set a STORGRP, and set NUNIT to a value greater than one. BrightStor CA-Allocate selects the first volume, and MVS selects any others. For those volumes MVS selects, PERMANENT data sets require that their volume(s) be mounted as STORAGE and TEMPORARY data sets require that their volume(s) be mounted as PUBLIC or STORAGE.

NVOL

Format type = numeric
Default = 1
Non-VSAM

The NVOL variable initially contains the number of volume serial numbers specified for the allocation. For example, if VOL=SER=(VOL001,VOL002) is specified in the JCL, NVOL contains a value of 2.

NVOL should only be changed when BrightStor CA-Allocate is redirecting the initial data set allocations, something that occurs in ALLOC and DEFINE for non-SMS-managed data sets only. When NVOL is set to a value greater than 1, in addition to the initial volume, BrightStor CA-Allocate selects NVOL-1 candidate volumes. Since no space is reserved on the candidate volumes, it is recommended that NVOL always be set to 1 when BrightStor CA-Allocate redirects. Then, when the data set needs to extend onto an additional volume, BrightStor CA-Allocate's End-Of-Volume Support selects the new volume. For NonVSAM, this avoids the often misleading "IEF283I ... NOT DELETED 8" messages that are issued when data sets with unused candidate volumes are deleted.

When redirection is requested by assigning a STORGRP in either ALLOC or DEFINE, NVOL defaults to a value of 1 unless otherwise specified. BrightStor CA-Allocate reduces any value specified for NVOL so that it does not exceed the number of volumes eligible as defined by the UNIT parameter or other ASR specifications.

Set NVOL to 0 to ignore all volumes specified by JCL and make the allocation request non-specific. MVS selects an appropriate volume based on the unit information provided.

Note: Changing the values in both the NVOL and VLCT variables for the same non-SMS-managed new data set allocation (that is redirected by BrightStor CA-Allocate) can cause unpredictable results.

NVOLD

Format type = numeric
VSAM Data Component

The NVOLD variable contains the number of user-requested volumes. It is obtained from the data component catalog entry. ASR can modify NVOLD in the DEFINE environment. A decrease in the original value is only allowed. No increases. Volumes in excess of the NVOLD are clipped from the Data component.

This works for SMS, non-SMS VSAM and JCL VSAM files. It is not dependent on redirection. If the value is set to zero it is changed to 1.

NVOLI

Format type = numeric
VSAM Index Component

The NVOLI variable contains the number of user-requested volumes. It is obtained from the index component catalog entry. ASR can modify NVOLI in the DEFINE environment. A decrease in the original value is only allowed. No increases. Volumes in excess of the NVOLI are clipped from the Index component.

This works for SMS, non-SMS VSAM and JCL VSAM files. It is not dependent on redirection. If the value is set to zero it is changed to 1.

N0 to N40

Format type = numeric
Maximum size = 2147483647
Non-VSAM and VSAM

The N0 through N40 variables are user numeric variables. They can be used as work variables as needed. As with the C0 through C40 variables, N0 through N40 are passed through to all User Exits.

OPTBLK

Format type = numeric
Non-VSAM

This variable contains the System Determined Blocksize calculated by the System when the JCL specifies BLKSIZE=0. DASD Calc Services is used to obtain this value. This service became available with DFP 3.1. It is an input-only variable that is available in the ALLOC and SPACE environments. In the SPACE environment, OPTBLK is calculated using the actual target device. In the ALLOC environment, OPTBLK is calculated using the UNITTYPE variable. It is available only when all the following conditions are true:

- XA or higher
- DFP 3.1 or higher
- RECFM specified
- LRECL specified

This variable can be used to set the &BLKSIZE variable to the System Determined Blocksize. If the &BLKSIZE variable is used to modify the block size, the reblock bit in the format-1 DSCB is not set. This can cause problems when moving from one device to another because the system does not automatically reblock. This problem can be overcome by always setting &BLKSIZE = &OPTBLK.

If the Block Size is zero (&BLKSIZE = 0), an internal call is made to the DASD Calc Services routine to get the System Determined Blocksize for space calculations.

OWNERID

Format type = character
Maximum size = 8
VSAM

The OWNERID variable contains the owner field from the Catalog Management base component control block.

PGM

Format type = character
Maximum size = 8
Non-VSAM and VSAM

The PGM variable contains the name of the program performing the allocation from the PGM= parameter of the job step.

For ACS, this value is established by the requestor of the processing ACS.

PGMRNAME

Format type = character
Maximum size = 20
Non-VSAM and VSAM

The PGMRNAME variable contains the contents of the programmer name field from the job card.

POOLSUB

Format type = character
Maximum size = 1
Non-VSAM and VSAM
SMS-Managed and non-SMS-Managed

NonSMS-Managed Allocations

&POOLSUB = 'Y' tells BrightStor CA-Allocate to automatically select a storage group based on volume serial number information specified for the original allocation. BrightStor CA-Allocate uses volume serial numbers found in &ALLVOL, &ALLVOLI, and &ALLVOLD to determine a storage group for the allocation.

POOLSUB assumes that either real volume serial numbers or storage group names exists in the volume serial field for the original allocation. When a storage group name is found as a volume serial number, that storage group name is assigned for the allocation. With POOLSUB, users can specify storage group names in JCL, TSO, and other allocation requests.

When a single real volume serial number is specified for the original allocation, storage group definitions are searched to find a storage group that includes the same volume. If one is found, that storage group name is set for the allocation. Instead of the data set being allocated to a specific volume serial number, it is allocated to one of the volumes in the related storage group.

POOLSUB processing sets a storage group only if &STORGRP, &STORGRPD, and &STORGRPI have not previously been set. POOLSUB automatically sets the &STORGRP, &STORGRPD, and &STORGRPI variables when setting a storage group name. If any of the above three storage group variables is explicitly defined in the ASR, POOLSUB processing is disabled.

If more than one volume serial number is found in the original allocation, up to four secondary storage groups can be specified for the allocation. All volumes chosen for the allocation are selected from the primary storage group if possible.

Note: If POOLSUB is unable to find a match, no substitution occurs and MVS attempts the allocation on the original volume(s) specified in the JCL.

SMS-Managed Allocations

Setting &POOLSUB = 'Y' has the same effect as setting &STORGRP = 'SMSEOV', which results in BrightStor CA-Allocate associating a candidate volume to the data set. For additional information, see the section **Using EOV and EOV_VSAM Environments** with a SMS-Managed Data Set in the chapter "**Concepts and Facilities**."

Use the following Storage Group definitions to evaluate the POOLSUB processing examples:

VOLUME SERIAL	FREESPACE THRESHOLD	STORAGE GROUP(S)	VOLUME SERIAL	FREESPACE THRESHOLD	STORAGE GROUP(S)
SSL80*	05	SSLDA	DMS902	01	DMS90B
SSL801	05	SSL80A	DMS901	01	DMS90A
SSL802	00	SSL80B	DMS903	01	DMS90C
SSL803	10	SSL80C	DMS9D*	01	DMS9DA
SSL804	00	SSL80D	DMS90*	01	DMS90X
SSL805	00	SSL80E			

EXAMPLE 1: non-VSAM

```
File   Edit   Confirm   Menu   Utilities   Compilers   Test   Help
------------------------------------------------------------------------
EDIT                        (IEFBR14) - 01.19                 Columns 00001 00072
Command ==> ____________________________________________________ Scroll ==> CSR

****** ************************** Top of  Data ********************************
000001  //STEP1    EXEC  PGM=IEFBR14
000002  //DD1       DD   DSN=AAA,DISP=(,CATLG),SPACE=(TRK,(1)),UNIT=SYSALLDA,
000003  //         VOL=SER= DMSK01
000004  //DD2       DD   DSN=BBB,DISP=(,CATLG),SPACE=(TRK,(1)),UNIT=SYSALLDA,
000005  //         VOL=SER= DMSK01,SSL801
000006  //DD3       DD   DSN=CCC,DISP=(,CATLG),SPACE=(TRK,(1)),UNIT=SYSALLDA,
000007  //         VOL=SER= SSLDA,SSL801,SSL802
000008  //DD4       DD   DSN=DDD,DISP=(,CATLG),SPACE=(TRK,(1)),UNIT=SYSALLDA
000009  //         VOL=SER= SSL801,DMS9DA,SSL803,SSL804
000010  //DD4       DD   DSN=EEE,DISP=(,CATLG),SPACE=(TRK,(1)),UNIT=SYSALLDA
000011  //         VOL=SER= SSL801,DMSK01,DMS902
****** ********************* Bottom  of  Data ********************************
```

POOLSUB matches volume serial numbers by searching the table from top to bottom and uses the first matching entry. Using the storage group definitions and the sample JCL shown above, POOLSUB sets &STORGRP for the each of the above DD statements as follows:

DD	&STORGRP	Reason
DD1	NULL	DMSK01 is not found in the table.
DD2	'SSLDA'	DMSK01 is not found and SSL801 matches SSL80*.
DD3	'SSLDA','SSLDA','SSLDA'	All three serial numbers match with SSL80*.
DD4	'SSLDA','DMS9DA',SSLDA', 'SSLDA', 'SSL801','SSL803', and 'SSL803'	All match SSL80*, and DMS9DA matches DMS9DA.
DD5	'SSLDA','DMS90A' 'SSL801'	matches with SSL80*, 'DMSK01' is not found in the table, and 'DMS901' matches with DMS90A

EXAMPLE 2: VSAM

```
File   Edit   Confirm   Menu   Utilities   Compilers   Test   Help
-------------------------------------------------------------------------
EDIT         SBDLM.UTIL.CNTL(IDCAMS) - 01.01                Columns 00001 00072
Command ==> _____________________________________________ Scroll ==> CSR

****** ************************** Top of  Data ********************************
000001  //STEP2    EXEC  PGM=IDCAMS
000002  //SYSPRINT  DD  SYSOUT=*
000003  //SYSIN     DD  *
000004  DEFINE  CL(NAME(TEST1.KSDS) -
000005          KEYS (19 4) RECORDSIZE(80 400) CYLINDERS(1 5) CISZ(512))
000006          DATA(NAME(TEST1.DATA) VOLUMES  DMS9DA * **  ) -
000007          INDEX(NAME(TEST1.INDEX) VOLUMES  DMS903 )
000008  DEFINE  CL(NAME(TEST2.KSDS) -
000009          KEYS (19 4) RECORDSIZE(80 400) CYLINDERS(1 5) CISZ(512))
000010          DATA(NAME(TEST2.DATA) VOLUMES  SSL801 SSL802  ) -
000011          INDEX(NAME(TEST2.INDEX) VOLUMES  SSL803 )
000012  DEFINE  CL(NAME(TEST3.KSDS) KEYS (19 4) RECORDSIZE(80 400) -
000013          CYLINDERS(1 5) VOLUMES  SSL801 SSL802 DMSK01 SSL804 -
000014          DATA(NAME(TEST3.DATA))
000015          INDEX(NAME(TEST3.INDEX))
****** ********************* Bottom  of  Data ********************************
```

POOLSUB sets STORGRP, STORGRPI, and STORGRPD as shown below:

- For the first DEFINE, POOLSUB sets:

```
&STORGRPD  = 'DMS9DA'
&STORGRPI  = 'DMS90C'
&STORGRP   = 'DMS9DA','DMS90C'.
```

- For the second DEFINE, POOLSUB sets:

```
&STORGRPD  = 'SSLDA','SSLDA'
&STORGRPI  = 'SSLDA'
&STORGRP   = 'SSLDA','SSLDA','SSLDA'.
```

- For the third DEFINE, POOLSUB sets:

```
&STORGRPD  = 'SSLDA','SSLDA','SSLDA'
&STORGRPI  = 'SSLDA','SSLDA','SSLDA'
&STORGRP   = 'SSLDA','SSLDA','SSLDA'
```

PRIMARY

Format type = numeric
Non-VSAM and VSAM

For non-VSAM, the PRIMARY variable starts out with the primary allocation quantity in units specified by the SPACTYPE variable. If the SPACTYPE is modified, the PRIMARY and SECONDARY variables are converted to the equivalent allocation in the new allocation unit. For non-VSAM, the PRIMARY variable can only be changed in the ALLOC and SPACE environments.

For VSAM, the PRIMARY variable starts out with the primary allocation quantity obtained from the base cluster catalog entry in units specified by the SPACTYPE variable. If the SPACTYPE is modified, the PRIMARY and SECONDARY variables are converted to the equivalent allocation in the new allocation unit.

For VSAM, the PRIMARY variable can only be changed in the DEFINE environment.

Caution: Because VSAM does not allow the setting of both cluster and component space attributes, neither can BrightStor CA-Allocate. If the DEFINE and the ASR interact to set both component and cluster space attributes, the ASR settings are ignored, and unexpected results are likely.

If the following ASR statements are used:

```
IF &PRIMARY = 0 THEN
  DO
   SET &PRIMARY = &PRIMARYD * 2
   SET &PRIMARYD = 0
   EXIT CODE(0)
  END
```

PRIMARY appears to be set, but it is not used to allocate space for the cluster. IDCAMS fails the allocation with the error IDC3009I 212-4.

Note: See the description of variable "SPACTYPE" for an example of how changing it affects the values of PRIMARY and SECONDARY.

Note: For SMS-managed allocations, BrightStor CA-Allocate's ability to modify this variable is contingent upon having access to the appropriate system control blocks.

See the associated variables PRIMARY, DC_PRIMARY, and see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: PRI

PRIMARYD

Format type = numeric
VSAM Data Component

The PRIMARYD variable starts out with the primary quantity obtained from the data component catalog entry in units specified by the SPACTYPED variable. If SPACTYPED is modified, the PRIMARYD and SECONDARYD variables are converted to the equivalent allocation in the new allocation unit. If the "Additional Volume Amount=SECONDARY" Date Class attribute is associated with the cluster, then IBM services will have changed this space amount to reflect the secondary amount the data component was originally allocated with.

Caution: Because VSAM does not allow the setting of both cluster and component space attributes, neither can BrightStor CA-Allocate. If the DEFINE and the ASR interact to set both component and cluster space attributes, the ASR settings are ignored, and unexpected results are likely.

If the following ASR statements are used:

```
IF &PRIMARYD = 0 THEN
  DO
   SET &PRIMARYD = &PRIMARY * 3
   SET &PRIMARY = 0
   EXIT CODE(0)
  END
```

PRIMARYD appears to be set, but it is not used to allocate space for the component. IDCAMS fails the allocation with the error IDC3009I 212-4.

The PRIMARYD variable can be changed only in the DEFINE and EOV_VSAM environments. See the description of variable "SPACTYPED" for an example of how changing this variable affects the values of PRIMARYD and SECONDARYD.

When the EOV_VSAM environment is entered, the value of the failing component's PRIMARY(D/I) and/or SECONDARY(D/I) allocation amount may be changed. Changes to the allocation amount are in effect only for the allocation of new extents while the data set is open; they are lost when the data set is closed and are not saved in the catalog.

The value in PRIMARYI is obtained from the index component catalog entry containing the index component it was initially created with. If an EOV condition occurs after the data set is opened BrightStor CA-Allocate is used to change this space amount (during an EXTEND Environment invocation, for example). PRIMARYI will no longer reflect the current primary allocation amount. The CURRENT_PRIMARYI variable will accurately reflect the current value. Please review this variable's description for information on how the any inconsistencies in these variables may impact a VSAM data component's ability to extend onto a new volume.

When a request for space on a new volume space is made by VSAM, that request is always based on the primary amount. Each additional request on an existing volume is based on the secondary amount. Because the EOV_VSAM environment allocates a new candidate to the data set, it must always use the primary amount in order to select a new volume.

If the allocation request is based on cylinders (and not a keyrange data set), the request is not considered to be a contiguous request and can be satisfied with up to 5 extents. If the allocation request is based on tracks (or a keyrange data set), the request is considered to be a contiguous request and must be satisfied in a single extent. This is a limitation of IDCAMS.

In the EOV_VSAM environment, only the allocation amount for the failing component can be changed. For example, if the data component has run out of space, setting the value of PRIMARYI (the index component) has no effect. Also, in order for the change to take effect, you must:

```
SET &STORGRP or &POOLSUB
EXIT CODE (0)
```

Finally, the SPACTYPED and SPACTYPEI fields can indirectly influence the value of these fields. For example, if SPACTYPED is changed from TRKS to CYLS, the primary and secondary amounts are changed to reflect their values in cylinders. This can be quite useful when coding an ASR in order to base decisions on a single set of rules. However, the SPACTYPE(n) field for the failing component must be changed back to its original specified value to enable the EOV_VSAM environment to change the value for VSAM. The allocation type cannot be permanently changed because VSAM bases its access on the type of allocation being used.

Alias: PDC

PRIMARYI **Format type = numeric**

VSAM Index Component

The PRIMARYI variable starts with the primary quantity obtained from the index component catalog entry in units specified by the SPACTYPEI variable. If SPACTYPEI is modified, the PRIMARYI and SECONDARYI variables are converted to the equivalent allocation in the new allocation unit. If the “Additional Volume.

Amount=SECONDARY” Date Class attribute is associated with the cluster, then IBM services will have changed this space amount to reflect what the secondary amount the index component was originally allocated with.

Caution: Because VSAM does not allow the setting of both cluster and component space attributes, neither can BrightStor CA-Allocate. If the DEFINE and the ASR interact to set both component and cluster space attributes, the ASR settings are ignored, and unexpected results are likely.

If the following ASR statements are used:

```
IF &PRIMARYI = 0 THEN
  DO
   SET &PRIMARYI = &PRIMARY * 3
   SET &PRIMARY = 0
   EXIT CODE(0)
  END
```

PRIMARYI appears to be set, but it is not used to allocate space for the component. IDCAMS fails the allocation with the error IDC3009I 212-4.

The value in PRIMARYI is obtained from the index component catalog entry containing the index component was initially created with. If between the time the data set was opened and it encounters an EOV condition, BrightStor CA-Allocate was used to change this space amount (during an EXTEND Environment invocation, for example), then PRIMARYI will no longer reflect the current primary allocation amount. The CURRENT_PRIMARYI variable will accurately reflect the current value. Please review this variable's description for information on how the any inconsistencies in these variables may impact a VSAM data component's ability to extend onto a new volume.

The PRIMARYI variable can be changed only in the DEFINE and EOV_VSAM environments. See the description of variable "SECONDARYI" for an example of how changing this variable affects the values of PRIMARYI and SECONDARYI.

Alias: PIC

PROCSTEP **Format type = character**
Maximum size = 8
Non-VSAM and VSAM

The PROCSTEP variable contains the name of the step that called the procedure. If a proc is not being executed in a batch job, this variable is blank.

REASON **Format type = numeric**
Default = 0
VSAM

The REASON variable is used to set the reason code in IBM message IDC3009I when the VDSPROG ASR returns a nonzero return code via EXIT CODE(n). Variable REASON specifies the 'crs' portion of the message below the reason code for the error. The ASR, like the IBM code, can use REASON to explain why it is failing the operation.

```
IDC3009I VSAM CATALOG RETURN CODE IS rc-REASON CODE IS IGGOCLaa-crs
```

RECFM

Format type = character
Maximum size = 5
Non-VSAM

The RECFM variable represents the record format of the data set. This variable can be changed in the ALLOC or SPACE environments to any valid combination of the 1 to 4 record format and characteristics attributes listed in the MVS JCL reference manual.

RECORG

Format type = character
Maximum size = 2
VSAM

The RECORG variable contains the VSAM record organization type obtained from the base cluster catalog entry. Valid values are shown below.

For ACS, this value is established by the requestor of the processing ACS.

Variable Value	Meaning	IDCAMS Terminology
KS	Key Sequenced	INDEXED
ES	Entry Sequenced	NONINDEXED
RR	Relative Record	NUMBERED
LL	Linear	LINEAR

REFDD

Format type = character
Maximum size = 27
Non-VSAM

This variable is applicable only if SMS is turned on for the system. It contains the REFDD coded in JCL. Because there are three forms of REFDD you must check the REFDDNQUAL variable to determine which form was coded. If REFDD is not blank and:

if &REFDDNQUAL = 1 then	&REFDD = *.ddname
	&REFDD(1) = ddname
if &REFDDNQUAL = 2 then	&REFDD = *.stepname.ddname
	&REFDD(1) = stepname
	&REFDD(2) = ddname

if &REFDDNQUAL = 3 then	&REFDD = *.stepname.procstepname.ddname &REFDD(1) = stepname &REFDD(2) = procstepname &REFDD(3) = ddname

REFDDNQUAL

Format type = numeric
Non-VSAM

This variable is applicable only if SMS is turned on for the system. It indicates the number of QUALIFIERS in the REFDD parameter. See REFDD variable above.

REPLICATE

Format type = character
Maximum size = 1
VSAM

The REPLICATE variable is a “yes” or “no” flag indicating whether or not REPLICATE was specified while defining a VSAM KSDS data set. Possible values are:

Y	Attribute was specified.
N	Attribute was not specified.

This variable can only be changed in the DEFINE environment. Possible values are:

Y	Add this attribute.
N	Remove this attribute.

Alias: REP

RETPD

Format type = numeric
Non-VSAM and VSAM

This modifiable variable contains the number of days until the data set expires.

For example, if today's date is July 1, 19xx and it is desired that a data set expire on the last day of July, 19xx, the RETPD variable can be changed as follows:

```
SET &RETPD = 30
```

See the description of variable "EXPDT." Note that the use of both EXPDT and RETPD in a VDSPROG ASR is not recommended. If both EXPDT and RETPD are assigned a new value in the VDSPROG ASR, the value assigned to EXPDT takes precedence over that assigned to RETPD.

For ACS, this value is established by the requestor of the processing ACS.

Note: For SMS-managed allocations, BrightStor CA-Allocate's ability to modify this variable is contingent upon having access to the appropriate system control blocks.

RLSE

Format type = character
Maximum size = 1
Non-VSAM

The RLSE variable indicates whether or not RLSE was specified in the SPACE parameter for the current allocation request. Possible values are:

Y	release unused space when the data set is closed
N	do not release unused space when the data set is closed

Note: For SMS-managed allocations, BrightStor CA-Allocate's ability to modify this variable is contingent upon having access to the appropriate system control blocks.

SECMODEL

Format type = character
Maximum size = 44
Non-VSAM

This variable contains the RACF profile specified in the DD parameter SECMODEL. It is only applicable on systems where the SMS address space is active.

SECMODG

Format type = character
Maximum size = 1
Non-VSAM

This variable indicates whether or not 'GENERIC' was specified as the RACF profile in the DD parameter SECMODEL. It is only applicable on systems where the SMS address space is active. Possible values are:

Y	"SECMODEL=GENERIC" was specified
N	"SECMODEL=GENERIC" was not specified

SECONDARY

Format type = numeric
Non-VSAM and VSAM

For non-VSAM, the SECONDARY variable starts out with the secondary allocation quantity in units specified by the SPACTYPE variable. If SPACTYPE is modified, the PRIMARY and SECONDARY variables are converted to the equivalent allocation in the new allocation unit. The secondary amount can be changed to any positive number or to zero to disallow secondary extents, or it can be increased at extend time to avoid running into the 16-extent limitation. Note, however, that if the secondary amount is changed, the space type is changed to blocks.

```
/* INCREASE SECONDARY IF WE'RE AT 9 OR HIGHER EXTENTS */
IF &VAMENVIR = 'EXTEND' THEN IF &EXTENTS > 8 THEN DO
    SET &SECONDARY = 2 * &SECONDARY
END
```

For VSAM, the SECONDARY starts out with the secondary quantity obtained from the base cluster catalog entry in units specified by the SPACTYPE variable. If SPACTYPE is modified, the PRIMARY and SECONDARY variables are converted to the equivalent allocation in the new allocation unit. Also, the SECONDARY variable can only be changed in the DEFINE environment.

Caution: Because VSAM does not allow the setting of both cluster and component space attributes, neither can BrightStor CA-Allocate. If the DEFINE and the ASR interact to set both component and cluster space attributes, the ASR settings is ignored, and unexpected results are likely.

If the following ASR statements are used:

```
IF &SECONDARY = 0 THEN
  DO
   SET &SECONDARY = &SECONDARYD * 2
   SET &SECONDARYD = 0
   EXIT CODE(0)
  END
```

SECONDARY appears to be set, but it is not used to allocate space for the cluster. IDCAMS fails the allocation with the error IDC3009I 212-4.

Note: See the description of variable "SPACTYPE" for an example of how changing this variable affects the values of SECONDARY and PRIMARY.

Note: For SMS-managed allocations, BrightStor CA-Allocate's ability to modify this variable is contingent upon having access to the appropriate system control blocks.

See the associated variables SECONDARY, DC_SECONDARY, and see the section **Enhanced DATACLAS Support** in the chapter "**Concepts and Facilities**" of this guide.

Alias: SEC

SECONDARYD

Format type = numeric
VSAM Data Component

The SECONDARYD variable starts out with the secondary allocation quantity obtained from the data component catalog entry in units specified by the SPACTYPED variable. If SPACTYPED is modified, the PRIMARYD and SECONDARYD variables are converted to the equivalent allocation in the new allocation unit. The SECONDARYD variable can only be changed in the DEFINE and EOV_VSAM environments.

Caution: Because VSAM does not allow the setting of both cluster and component space attributes, neither can BrightStor CA-Allocate. If the DEFINE and the ASR interact to set both component and cluster space attributes, the ASR settings is ignored, and unexpected results are likely.

If the following ASR statements are used:

```
IF &SECONDARYD = 0 THEN
  DO
   SET &SECONDARYD = &SECONDARY * 3
   SET &SECONDARY = 0
   EXIT CODE(0)
  END
```

SECONDARYD appears to be set, but it is not used to allocate space for the component. IDCAMS fails the allocation with the error IDC3009I 212-4.

Note: Refer to "SPACTYPED" for an example of how changing this variable affects the values of PRIMARYD and SECONDARYD.

When the EOV_VSAM environment is entered, the value of the failing component's PRIMARY(D/I) or SECONDARY(D/I) allocation amount may be changed. Changes to the allocation amount are in affect only for the allocation of new extents while the data set is open; they are lost when the data set is closed and are not saved in the catalog.

The value in SECONDARYD is obtained from the data component catalog entry that contains the data component initial creation information. If an EOV condition is encountered after the data set was opened, BrightStor CA-Allocate is used to change the space amount (during an EXTEND Environment invocation, for example).

SECONDARYD will no longer reflect the current primary allocation amount. The CURRENT_SECONDARYD variable will accurately reflect the current value. Please review this variable's description for information on how the any inconsistencies in these variables may impact a VSAM data component's ability to extend onto a new volume.

When a request for space on a new volume is made by VSAM, that request is always based on the primary amount. Each additional request on an existing volume is based on the secondary amount. Because the EOV_VSAM environment allocates a new candidate volume to the data set, it needs to make its selection based on the primary allocation quantity. If the allocation request is based on cylinders (and not a keyrange data set), the request is not considered to be a contiguous request and can be satisfied with up to 5 extents. If the allocation request is based on tracks (or a keyrange data set), the request is considered to be a contiguous request and must be satisfied in a single extent. These are limitations of IDCAMS.

Note: You can change the allocation amount only for the failing component. For example, if the data component has run out of space, setting the value of PRIMARYI (the INDEX component) has no effect. Also, in order for the change to take effect, you must:

```
SET &STORGRP or &POOLSUB
EXIT CODE (0)
```

Finally, the SPACTYPED and SPACTYPEI fields can indirectly influence the value of these fields. For example, if SPACTYPED is changed from TRKS to CYLS, the primary and secondary amounts are changed to reflect their values in cylinders. This can be quite useful when coding your ASR in order to base your decisions on a single set of rules. However, the SPACTYPE(n) field for the failing component must be changed back to its original specified value to enable the EOV_VSAM environment to change the value for VSAM. The allocation type cannot be permanently changed because VSAM bases its access on the type of allocation being used.

In the VSAM Extend environment, if a change is made to &SECONDARYD, this change will stay in effect while this cluster remains open. Depending upon how the ASR is coded, this could lead to a larger than expected value.

For example, if in the ASR &SECONDARYD is multiplied by a value for each pass through Extend:

```
SET &SECONDARYD = &SECONDARYD * 2
```

If the original value of &SECONDARYD was 4, after the first time through Extend, &SECONDARYD will now be 8. If Extend is entered again on this same volume and the above code is exercised again, &SECONDARYD will now be 16, and so on.

Alias: SDC

SECONDARYI

Format type = numeric
VSAM Index Component

The SECONDARYI variable starts out with the secondary quantity obtained from the index component catalog entry in units specified by the SPACTYPEI variable. If SPACTYPEI is modified, the PRIMARYI and SECONDARYI variables are converted to the equivalent allocation in the new allocation unit. The SECONDARYI variable can only be changed in the DEFINE and EOV_VSAM environments.

Caution: Because VSAM does not allow the setting of both cluster and component space attributes, neither can BrightStor CA-Allocate. If the DEFINE and the ASR interact to set both component and cluster space attributes, the ASR settings is ignored, and unexpected results are likely.

If the following ASR statements are used:

```
IF &SECONDARYI = 0 THEN
  DO
   SET &SECONDARYI = &SECONDARY * 3
   SET &SECONDARY = 0
   EXIT CODE(0)
  END
```

SECONDARYI appears to be set, but it is not used to allocate space for the component. IDCAMS fails the allocation with the error IDC3009I 212-4.

The value in SECONDARYI is obtained from the data component catalog entry that contains the data component initial creation information. If an EOV condition is encountered after the data set was opened, BrightStor CA-Allocate is used to change the space amount (during an EXTEND Environment invocation, for example).

SECONDARYI no longer reflects the current primary allocation amount. The CURRENT_SECONDARYI variable will accurately reflect the current value. Please review this variable's description for information on how the any inconsistencies in these variables may impact a VSAM data component's ability to extend onto a new volume.

See the description of variable "SPACTYPEI" for an example of how changing this variable affects the values of PRIMARYI and SECONDARYI.

In the VSAM Extend environment, if a change is made to &SECONDARYI, this change will stay in effect while this cluster remains open. Depending upon how the ASR is coded, this could lead to a larger than expected value.

For example, if in the ASR &SECONDARYI is multiplied by a value for each pass through Extend:

```
SET &SECONDARYI = &SECONDARYI * 2
```

If the original value of &SECONDARYI was 4, after the first time through Extend, &SECONDARYI will now be 8. If Extend is entered again on this same volume and the above code is exercised again, &SECONDARYI will now be 16, and so on.

Alias: SIC

SECONDARY_HAD_ZERO **Non-VSAM**
Format type = character
Maximum size = 1

The SECONDARY_HAD_ZERO variable indicates whether a data set has had its '0' (ZERO, Z) secondary amount (SEC) changed to '1' in order to avoid a D37 abend. This variable is only applicable with NonVSAM data sets in the EXTEND and EOV Environments when running in a PLSZSEC (Y) configuration.

Possible values are:

Y	Secondary was zero
N	Secondary was non-zero or, for SMS-Managed data sets, Z SEC was previously changed to NZ
?	Not running in a PLSZSEC (Y) configuration

This information can be used by the ASR in the EXTEND and EOV Environments to differentiate between those data sets that had a Z SEC amount from those that had a NZ one.

NonSMS-Managed data sets with Z SEC require the "Z SEC" Support provided by a PLSZSEC (Y) configuration for all extents and all EOV conditions within a step. SMS-Managed data sets with Z SEC only require it within a step up until the time of the second extent following an EOV condition.

See the descriptions of the PLSZSEC parameter earlier in this chapter and the **EOV Support for Zero Secondary** and the **Non-VSAM Processing in EXTEND** sections in the chapter "Concepts and Facilities" of this guide.

Alias: HAD_ZSEC, ZSEC

SECONDARY_ORIG **Format type = numeric**
Non-VSAM

The SECONDARY_ORIG variable contains the original secondary allocation quantity in units specified by the SPACTYPE variable. It can be used to reset the secondary allocation amount (SECONDARY, SEC) to its original value in either the EXTEND or EOV Environments.

In order to activate this variable for usage in the EXTEND or EOV environments, it needs to be referenced in the ACS environment for SMS files and in the ALLOC environment for nonSMS files.

See the descriptions of the "SECONDARY" and "SPACTYPE" variables in this chapter.

Alias: SEC_ORIG

SHRINKTABLE **Format type = character**
Maximum size = 8
VSAM

This variable provides an interface with BrightStor CA-Compress Data Compression (BrightStor CA-Compress), a data compression system available from Computer Associates as a separate system software product.

The SHRINKTABLE variable is used to specify the BrightStor CA-Compress compression algorithm or File Descriptor Table (FDT) name. Setting this variable indicates that compression is to be implemented on this data set. Refer to the BrightStor CA-Compress documentation for further information on its SUBSYS facility.

See also the description of the variable "SHRINKTASK."

SHRINKTASK

Format type = character
Maximum size = 4
Default = 'ZSAM'
VSAM

The SHRINKTASK variable is used to provide an interface with BrightStor CA-Compress, a data compression system available from Computer Associates as a separate system software product.

SHRINKTASK is used to specify the BrightStor CA-Compress Subsystem name. BrightStor CA-Allocate uses this Subsystem name when determining whether BrightStor CA-Compress is active on the system. Refer to your BrightStor CA-Compress documentation for further information on its SUBSYS facility.

See also the description of the variable "SHRINKTASK."

SIZE

Format type = numeric
Non-VSAM or VSAM
SMS-Managed or non-SMS-Managed

The SIZE variable contains the primary space amount in megabytes (MB) requested for a new data set allocation.

This variable returns the same value as the SIZEMB variable.

SIZEKB

Format type = numeric
Non-VSAM or VSAM
SMS-Managed or non-SMS-Managed

The SIZEKB variable contains the primary space amount in kilobytes (KB) requested for a new data set allocation, rounded up to the next KB.

SIZEMB

Format type = numeric
Non-VSAM or VSAM
SMS-Managed or non-SMS-Managed

The SIZEMB variable contains the primary space amount in megabytes (MB) requested for a new data set allocation, rounded up to the next MB. This variable returns the same value as the &SIZE variable.

SPACCVT

Format type = character
Maximum size = 1
Default = 'Y'
Non-VSAM or VSAM

The SPACCVT variable determines whether or not a space type conversion should be done when the SPACTYPE, SPACTYPED, or SPACTYPEI variables are modified. Possible values are:

Y	Honor the space type conversion request
N	Ignore the space type conversion request

The default value is 'Y', which triggers the "real time conversion" of the allocation quantities to the equivalent amount of the new allocation unit. To disable the otherwise automatic conversion, set SPACCVT to 'N'. For additional information, see the section **&UNIT** in the chapter "**Concepts and Facilities**" and the variable description of "SPACTYPE."

The flexibility described in the above paragraph has some special considerations. After the SPACCVT is modified to 'N', the ASR cannot examine the space parameters with converted information, so modifying SPACTYPE without modifying the other related variables results in a different allocation than expected.

Consider what happens in the examples below.

```
    SET &SPACCVT = 'N'
    SET &SPACTYPE = 'TRK'
```

If the space requested in the JCL is SPACE=(CYL,(100,10)) then the above ASR statements cause the allocation to become SPACE=(TRK,(100,10)). Note that if SPACCVT = 'Y' then the allocation becomes SPACE=(TRK,(1500,150)), assuming a 3380 device, because the PRIMARY and SECONDARY were converted.

If the space requested in the JCL is space=(CYL,(25,10) then the ASR statements below cause the allocation to become SPACE=(CYL,(100,10)) because the PRIMARY and SECONDARY amounts are unchanged even though the SPACTYPE changes. Note that if SPACCVT = 'Y' were selected, then the allocation would remain SPACE=(CYL,(25,10)).

```
    SET &SPACCVT = 'N'
    SET &C0 = &SPACTYPE
    SET &SPACTYPE = 'TRK'
    SET &PRIMARY = &PRIMARY
    SET &SECONDARY = &SECONDARY
 IF  &PRIMARY LT 100 THEN
   DO
    SET &PRIMARY = 100
   END
    SET &SPACTYPE = &C0
    SET &PRIMARY = &PRIMARY
    SET &SECONDARY = &SECONDARY
```

SPACTYPE

Format type = character
Maximum size = 3
Default = null
Non-VSAM or VSAM

The SPACTYPE variable initially contains the type of space being requested. This variable can be modified by the Allocation Selection Routine. Modifying SPACTYPE, depending on the value of the SPACCVT variable, may result in the allocation quantities being converted based on the allocation unit. A general overview of this process can be found in the section **&UNIT** in the chapter "**Concepts and Facilities**." Details on how the value of SPACCVT affects this process can be found in description of variable "SPACCVT."

For Non-VSAM allocations, this is retrieved from the SPACE parameter, and can be one of four possible values:

Value	Meaning
AVR	average records request
BLK	number of blocks
CYL	number of cylinders
TRK	number of tracks

Six types of conversions are supported for Non-VSAM allocation:

FROM	TO
AVR	BLK
BLK	TRK
CYL	BLK
CYL	TRK
TRK	BLK
TRK	CYL

For example, if the JCL specifies "SPACE=(TRK,(16,16))", and the SPACCVT default value of 'Y' is used, then the following series of ASR statements converts that allocation request to "SPACE=(CYL,(2,2))":

```
SET &UNITTYPE    = '3380'
SET &SPACTYPE    = 'CYL'
SET &PRIMARY     = &PRIMARY
SET &SECONDARY   = &SECONDARY
```

Note the rounding that occurred. When the allocations are converted, they are "rounded-up" to the higher quantity. The objective in this is never to end up with a reduced allocation amount as a result of a change in SPACTYPE. This loss of precision needs to be considered before proceeding with space type conversions.

Conversions between AVR, BLK, CYL, and TRK are influenced by the value of the UNITTYPE variable, and where applicable, AVGRECTYPE, AVGRECSIZE, AVGBLK, BLKSIZE, and OPTBLK. BrightStor CA-Allocate has internal tables containing device characteristics that it uses when it does space type conversions. For more information, see the description of variable "UNITTYPE."

SPACTYPE = 'AVR' indicates that the allocation is in units of average record size. Access to the average record size value is provided through the AVGRECSIZE variable and the factor type value through AVGRECTYPE. Figure 2-18, "Sample Enhanced DATACLAS ASR," contains some ASR statements used in converting an allocation to average records.

For VSAM allocations, information about the allocation unit is retrieved from the base cluster catalog entry, and can be one of five possible values:

Value	Meaning
CYL	number of cylinders
KB	number of kilobytes
MB	number of megabytes
REC	number of records
TRK	number of tracks

The following three conversions are supported for VSAM allocations:

FROM	TO
CYL	TRK
KB	MB
TRK	CYL

Rounding, loss of precision, and the role of the UNITTYPE and SPACCVT variables must be considered for VSAM, as well as non-VSAM.

Note: For SMS-managed allocations, BrightStor CA-Allocate's ability to modify this variable is contingent upon having access to the appropriate system control blocks.

If you are changing SPACTYPE from BLKS to TRKS, you must use the following in your ASR:

```
SET &BLKSIZE=&AVGBLK
```

Because BrightStor CA-Allocate depends on the accuracy of the &BLKSIZE variable, if you do not use the above, BrightStor CA-Allocate incorrectly computes the new space amount, resulting in errors.

Alias: STY

SPACTYPED **Format type = character**
Maximum size = 3
Default = null
VSAM

The SPACTYPED variable initially contains the type of space being requested, and is obtained from the data component catalog entry. It is modified in the same way as the SPACTYPE variable for VSAM.

For example, if the DEFINE of a data component specified “TRACK(16,16)”, the following series of ASR statements, if the SPACCVT default value of 'Y' is used, converts that allocation request to “CYLINDERS(2,2)”:

```
SET &UNITTYPED   = '3380'
SET &SPACTYPED   = 'CYL'
SET &PRIMARYD    = &PRIMARYD
SET &SECONDARYD  = &SECONDARYD
```

The same information given above regarding the rounding, loss of precision, and the role of the UNITTYPED and SPACCVT variables in regards to non-VSAM space type conversions applies as well here to VSAM ones.

Alias: STYD

SPACTYPEI

Format type = character
Maximum size = 3
Default = null
VSAM

The SPACTYPEI variable initially contains the type of space being requested, and is obtained from the index component catalog entry. It is modified in the same way as the SPACTYPE variable for VSAM.

For example, if the DEFINE of an index component specified “TRACK(16 16)”, the following series of ASR statements, if the SPACCVT default value of 'Y' is used, converts that allocation request to “CYLINDER(2 2)”:

```
SET &UNITTYPEI   = '3380'
SET &SPACTYPEI   = 'CYL'
SET &PRIMARYI    = &PRIMARYI
SET &SECONDARYI  = &SECONDARYI
```

The same information given above regarding the rounding, loss of precision, and the role of the UNITTYPEI and SPACCVT variables in regards to non-VSAM space type conversions applies as well here to VSAM ones.

Alias: STYI

STEPCATOK

Format type = character
Maximum size = 1
Default = ’Y’
VSAM

The STEPCATOK variable determines whether to use (STEPCATOK=Y) or disallow (STEPCATOK=N) the catalog allocated through STEPCAT. If the job step being processed does not have a STEPCAT DD statement allocated to it, changing this variable has no effect on the DEFINE.

This variable can be used in conjunction with variables CATONDEFOK and JOBCATOK to enforce installation standards for catalog use, as shown in this example:

```
SET &CATONDEFOK  = 'N'
SET &STEPCATOK   = 'N'
SET &JOBCATOK    = 'N'
WRITE 'The proper catalog to use is &CATDSN'
```

If STEPCATOK changes, the values of CATDSN, CATNQUAL, CATHLQ, and CATLLQ can also change. See also the descriptions of the CATDSN, CATNQUAL, CATONDEFOK, JOBCATOK, CATHLQ, and CATLLQ variables.

STEPNAME

Format type = character
Maximum size = 8
Non-VSAM or VSAM

The STEPNAME variable contains the name of the step that is executing. If the step executes a proc, STEPNAME is the proc step name. For the following JCL:

```
//PROC    PROC
//BR14    EXEC IEFBR14
// PEND
//A       EXEC PROC
```

STEPNAME is BR14 and the PROCSTEP variable is A.

STOP_NOT_CATLG2

Format type = character
Maximum size = 1
Default = 'N'
Non-VSAM
SMS-Managed or non-SMS-Managed

The STOP_NOT_CATLG2 variable is used in the SPACE environment during allocation of a new Non-VSAM data set. If an existing data set is already cataloged by the same DSNAME, then one of the following values can be specified:

N	Leaves the existing catalog entry unaltered
R	Renames the existing data set
D	Deletes and uncatalogs the existing data set
U	Uncatalogs the existing non-SMS-managed data set

SNC2 support does not support deleting, renaming, or uncataloging SMS managed data sets with candidate volumes.

Note: The &SNC2RC must be referenced after this variable in the ASR in order for this variable to be activated.

Alias: SNC2

STOP_NOT_CATLG2_NODE

Format type = character
Maximum size = 8
Default = 'SNC2'
Non-VSAM

The STOP_NOT_CATLG2_NODE variable is used in the SPACE environment during allocation of a new non-VSAM data set. If the same DSNAME is already cataloged and &SNC2 is set to 'R', then BrightStor CA-Allocate appends the value specified for this variable as the new LLQ.

Note: The &SNC2RC must be referenced after this variable in the ASR in order for this variable to be activated.

Alias: SNC2N

STOP_NOT_CATLG2_RC

Format type = numeric
Non-VSAM

This variable is used in the SPACE environment during allocation of a new Non-VSAM data set. If the data set name is already cataloged, it contains the Return Code from the function performed.

If the value of this variable is `0', then the function performed was successful and the new data set can be allocated, thus avoiding the NOT CATLGD2 condition. If this variable contains a NON-ZERO value, then the function failed with one of the following:

TABLE 4-5 SNC2RC Non-Zero Code and their Meanings

CODE	Meaning
1	The UNCATALOG option was chosen for an existing SMS-managed data set. This function cannot be performed against an SMS-managed data set.
2	The data set being allocated was not redirected by BrightStor CA-Allocate (for example, STORGRP was not set in the ALLOC environment). This function cannot be performed unless BrightStor CA-Allocate redirects the data set.
3	The existing data set has more than 20 volumes allocated to it. This function only supports data sets cataloged to no more than 20 volumes.
4	The UNCATALOG DELETE option failed. Refer to the BrightStor CA-Allocate and / or IBM diagnostics for problem determination.
5	The DELETE option failed. Refer to the BrightStor CA-Allocate and / or IBM diagnostics for problem determination.

CODE	Meaning
6	The UNCATALOG option was chosen for a new data set being allocated to the same volume as the existing data set. The new data set can not be allocated on this volume.
7	The existing data set is cataloged to a pseudo-volser. See VKGPARMS parameter "PLSARCVL" in this chapter for details. Since the existing data set is archived, the function was not performed.
8	A non-zero return code was returned by IBM macro "UCBLOOK." Refer to the BrightStor CA-Allocate and / or IBM diagnostics for problem determination.
9	See the meaning for code "8" above.
10	The IBM "UCBSCAN" service failed. Refer to the BrightStor CA-Allocate and / or IBM diagnostics for problem determination.
11	During the RENAME function, BrightStor CA-Allocate determined that the length of the new data set name (for example, length of original data set name plus length of new node plus one) was greater than 44 characters.
12	During the RENAME function, BrightStor CA-Allocate discovered that the new data set name was already cataloged.
13	The RENAME of the existing data set failed. Refer to the BrightStor CA-Allocate and / or IBM diagnostics for problem determination.
14	The new data set is being allocated from a TSO session. This function cannot be performed under TSO.
15	"D", "N", "R", or "U" not specified for &STOP_NOT_CATLG2.
16	An attempt was made to invoke the STOP NOT CATLG2 support from an environment in which it is not supported.
17	&SNC2N does not begin with alpha or national character.
18	Delete of an SMS managed data set failed when &SNC2 was specified with 'D'.
19	An attempt was made to invoke the STOP NOT CATLG2 support when IFCAT was not 'Y'.
20	RENAME or DELETE not allowed with TAPE data sets.

Alias: SNC2RC

STORCLAS

Format type = character
Maximum size = 8
Non-VSAM or VSAM
SMS-Managed only

STORCLAS contains the SMS storage class construct name. The original value in STORCLAS comes from the original user request, and any IBM ACS processing. In the ACS environment, the value can be set to any valid storage class name, or removed by setting it to null.

Setting a storage class makes a data set SMS-managed. Removing the storage class makes a data set non-SMS-managed.

The Storage Class name may not always be available in the DEFINE, SCRATCH, and SPACE environments or in the QUOTA environment when QUOTAFUNC=SCRATCH.

Alias: STORAGECLASS or SC

STORGRP

Format type = character(list)
Maximum size = 134
Non-VSAM and VSAM

STORGRP is one of several internal variables valid only for the current invocation of BrightStor CA-Allocate. It is by far the most powerful because of how it influences which volume is selected for a new data set allocation, and whether any volume at all is selected for an old data set encountering an out-of-space condition. The STORGRP variable is applicable to both SMS-managed and non-SMS-managed allocations.

Note: Refer to the descriptions of NVOL and UNIT for information on how their values determine the number of volumes selected for new data set allocations.

NonSMS-Managed Allocations

For new non-SMS-managed allocations, STORGRP is used to indicate which storage group BrightStor CA-Allocate is to pick a volume from. This storage group has to have been defined in the PDS member referenced by the PLSSTRMB parameter in the VKGPARMS member of PARMLIB when BrightStor CA-Allocate was brought up, or by the STORGRP parameter used the last time it was REFRESHed. For additional information, see the section Creating a Storage Group Definition in this chapter.

For NonVSAM allocations, whether to have BrightStor CA-Allocate select the volume(s) is decided in the ALLOC environment. For VSAM allocations, this is done in the DEFINE environment. STORGRP is used for volume selection at the VSAM cluster level. At the component level, STORGRPD and STORGRPI are used. When all three variables are used, the storage group specified for the individual component takes precedence over the one specified for the cluster. For additional information about volume selection at the VSAM component level, refer to the descriptions of STORGRPD and STORGRPI.

The STORGRP variable is initially set to null. It may be set to one or more storage groups, up to a maximum length of 134 characters counting commas but not the quotation marks. For example, consider the following statement that makes two storage groups available for an allocation:

```
SET &STORGRP = 'PRIMARY','SECONDRY'
```

In the above example, the 'PRIMARY' storage group is searched first for an eligible volume. It only searches 'SECONDRY' if it fails to find a suitable volume in 'PRIMARY'. For additional information on this process, see the section **Volume Selection Processing Logic** in the chapter "**Concepts and Facilities**."

As delivered, if BrightStor CA-Allocate is unable to select a volume for a new data set, it puts the original UNIT and VOL=SER information back and let the operating system attempt to make the allocation. The FAILIFNOVOLSEL variable can be used to override this default action and immediately fail the allocation if BrightStor CA-Allocate is unable to select a volume. For additional information on this process, see the description of variable "FAILIFNOVOLSEL."

For old data sets, the STORGRP variable can to used to recover from an end-of-volume condition, even if BrightStor CA-Allocate was not involved in the original allocation that there are separate end-of-volume environments for NonVSAM and VSAM data sets, EOV and EOV_VSAM, respectively, recovery from the condition is identical: instruct BrightStor CA-Allocate to pick an additional volume by either setting the STORGRP variable to the name of a valid storage group, or set the POOLSUB variable to 'Y'. For additional information, see the description of variable "POOLSUB."

In either case, if a volume is found that can support the allocation quantity, it is added to that data set's catalog entry and the extension request is re-driven. If no volume can be selected, then the extent request fails with errors indicating the original out-of-space condition. See the section **Technical Overview** in the chapter "**Concepts and Facilities**" for more information.

For the QUOTA, QSCAN, and QREBUILD environments, see the description of variable "STORGRP."

To have random volume selection when setting a storage group (&SG) add PLSOPT20 (Y) in the VKGPARMS member of PARMLIB. For further information on PLSOPT20, see PLSOPT20 in the section VKGPARMS Parameters at the beginning of this chapter.

SMS-Managed Allocations

For new SMS-managed allocations, BrightStor CA-Allocate does not directly select volumes. It simply tells SMS which ISMF-defined storage group to use. This is done in the ACS environment, where the STORGRP variable initially contains the name of the ISMF storage group that has either been set in the SMS ACS routines, or in the JCL or IDCAMS DEFINE statements used by the user requesting the allocation.

For the ALLOC, DEFINE, and SPACE environments, this information may not always be available. For additional information, see the section **ACS Environment** in the chapter "**Concepts and Facilities**."

For old SMS-managed allocations encountering an out-of-space condition, the STORGRP variable can be used to direct SMS to allocate an additional volume to the data set. Except that there are separate end-of-volume environments for NonVSAM and VSAM data sets, EOV and EOV_VSAM, respectively, recovery from the condition is identical: set the STORGRP variable to the keyword 'SMSEOV'. This instructs SMS to select another volume from the ISMF storage group that was associated with the data set when it was originally allocated. For additional information, see the section **Using EOV and EOV_VSAM Environments** with a SMS-Managed Data Set in the chapter "**Concepts and Facilities**."

Alias: STORAGEGROUP or SG

STORGRPD **Format type = character(list)**
Maximum size = 56
VSAM

VSAM data sets consist of one or two separate physical components (DATA and INDEX) logically associated together (CLUSTER). When volume selection is to be done at the cluster level, the STORGRP variable is used. When it is to be done separately for the data component, the STORGRPD variable is used. Except that the length of STORGRPD, counting commas but not the quotation marks, cannot exceed 56 characters, the function and use of this variable is identical to what is detailed for "STORGRP."

The storage group specified for STORGRPD is given preference over any specified for STORGRP when selecting the volume(s) for a data component. If the only storage group specified was for the STORGRPI variable, then a volume from there is selected for the data component.

Note: See the descriptions of the FAILIFNOVOLSEL, INDEXSEP, NVOLD, POOLSUB, and UNITD variables for information on how their values can influence what BrightStor CA-Allocate does with STORGRPD.

STORGRPI

Format type = character(list)
Maximum size = 56
VSAM

VSAM data sets consist of one or two separate physical components (DATA and INDEX) logically associated together (CLUSTER). When volume selection is to be done at the cluster level, the STORGRP variable is used. When it is to be done separately for the index component, the STORGRPI variable is used. The function and use of this variable is identical to what is detailed for "STORGRPD."

The storage group specified for STORGRPI is given preference over any specified for STORGRP when selecting a volume for the index component. If the only storage group specified was for the STORGRPD variable, then a volume from there is selected for the index component.

Note: See the descriptions of the FAILIFNOVOLSEL, INDEXSEP, NVOLI, POOLSUB, and UNITD variables elsewhere in this section for information on how their values can influence what BrightStor CA-Allocate does with STORGRPI.

SYSID

Format type = character
Maximum size = 4
Non-VSAM and VSAM

The SYSID variable contains the SMF identifier of the system on which BrightStor CA-Allocate is running.

TIME

Format type = character
Maximum size = 8
Non-VSAM and VSAM

The TIME variable contains the formatted system time at the moment the ASR statement containing it is executed. The time is in the format HH:MM:SS where HH=hours (01 to 24), MM=minutes, and SS=seconds.

TODAY

Format type = numeric
Non-VSAM and VSAM

The TODAY variable contains today's date in Julian format.

For example:

```
WRITE 'TODAY IS &TODAY'
```

On January 1, 2000, the above example output 'TODAY IS 2000001' and on October 1, 1998, it is 'TODAY is 1998274'.

UNIT

Format type = character
Maximum size = 8
Non-VSAM and VSAM

The UNIT variable initially contains the name of the unit that the user requested. It can be changed in the ALLOC and DEFINE environments to another unit name, such as SYSALLDA:

```
SET &UNIT='SYSALLDA'
```

If the unit chosen restricts the volumes, the storage group selected must have intersecting volumes, so that a volume can be selected. If the STORGRP chosen contains both '3380' and '3390' devices and UNIT is set to '3390', selection is only made from the '3390' volumes. Setting UNIT to 'SYSALLDA' makes both device types eligible because all disk volumes belong to 'SYSALLDA'. Also see the description of variable "STORGRP."

For non-VSAM, the default value is a null string.

For ACS, this value is established by the requestor of the processing ACS.

For VSAM allocations, the UNIT variable is used to assign units to the data or index component data sets if UNITD or UNITI are not specified.

UNIT variable changes in the ALLOC environment that require a change in the device class result in a device-type conversion. Possible device-type conversions are TAPE-to-DISK and DISK-to-TAPE. Depending on the conversion type, BrightStor CA-Allocate internally overrides the original values of the CMP3480, FILESEQ, LABEL, UNITAFF, VREFDDN, VREFDSN, and VREFSTP variables.

A detailed description of device-type conversions can be found in the section **Device-Type Conversions** in the chapter "**Concepts and Facilities**."

UNITAFF

Format type = character
Maximum size = 1
Non-VSAM

The UNITAFF variable indicates whether or not a unit affinity was specified for the current allocation request.

Possible values are:

Y	UNIT=AFF=ddname was specified
N	no unit affinity was specified

This variable can only be changed in the ALLOC Environment. The single possible output value is:

N disable any unit affinity that was specified

When a disk allocation is redirected, any unit affinity that was originally specified for the allocation is automatically disabled. A detailed explanation for this can be found in “Volume Referbacks / Unit Affinity and Disk-only Allocations.”

Any unit affinity is also disabled for TAPE-to-DISK and DISK-to-TAPE device-type conversions, but not for TAPE-to-TAPE allocation changes. A detailed description of why different actions are taken, depending on the conversion type, can be found in the section **Device-Type Conversions** in the chapter ”**Concepts and Facilities**.”

UNITD

Format type = character
Maximum size = 8
VSAM Data Component

The UNITD variable initially contains the name of the unit that the user requested. You can set it to another unit name, such as SYSALLDA.

```
SET &UNITD='SYSALLDA'
```

The unit name can be changed only in the DEFINE environment. If the selected unit restricts the volumes, the storage group selected should have intersecting volumes, or no volume can be selected. Setting UNITD to 'SYSALLDA' makes any disk device type eligible for selection because all disk volumes belong to 'SYSALLDA'. See also the description of variable "STORGRPD."

If the UNITD variable is not specified, the UNIT variable is used to assign a unit to the data component data set. If the UNIT variable is also not specified, the UNITI variable is used to assign volumes to the data component data set. If none of the three variables (UNIT, UNITD, or UNITI) is specified, UNITD defaults to SYSALLDA.

UNITI

Format type = character
Maximum size = 8
VSAM Index Component

The UNITI variable initially contains the name of the unit that the user requested. You can set it to another unit name, such as SYSALLDA.

```
SET &UNITI='SYSALLDA'
```

The unit name can be changed only in the DEFINE environment. If the selected unit restricts the volumes, the storage group selected should have intersecting volumes, or no volume can be selected. Setting UNITI to 'SYSALLDA' makes any disk device type eligible for selection because all disk volumes belong to 'SYSALLDA'. See also the description of variable "STORGRPI."

If the UNITI variable is not specified, the UNIT variable is used to assign a unit to the data component data set.

If the UNIT variable is also not specified, the UNITD variable is used to assign volumes to the index component data set. If none of the three variables (UNIT, UNITD, or UNITI) is specified, UNITI defaults to SYSALLDA.

UNITMISMATCH

Format type = character
Maximum size = 1
Non-VSAM
Non-SMS-Managed

The UNITMISMATCH variable is a “yes” or “no” flag indicating whether or not an inconsistency has been detected between what was specified in the JCL and what is in the catalog entry. Possible values are:

Y	catalog versus JCL inconsistency found
N	no inconsistency found.

When the UMM variable indicates 'Y' a JCL error occurs if the information in the UNIT variable is not corrected. The FIXUNITMISMATCH variable contains the correct information.

For more information about how this mismatch condition can arise, and how to correct it, see also the section **OLD Environment** in the chapter “**Concepts and Facilities**” and the description of variable “**FIXUNITMISMATCH**.”

Alias: UMM

UNITTYPE

Format type = character
Maximum size = 8
Default = '3380'
Non-VSAM and VSAM

The UNITTYPE variable contains a string value that specifies the type of unit for space type conversions involving either Non-VSAM or VSAM cluster allocations. The table below lists the valid values and device characteristics of each:

UNITTYPE	TRK/CYL
3340	12
3350	30
3375	12
3380	15
3390	15
9340	15

The UNITTYPE variable contains the unit type found in the UCB passed to the DADSM exit (that is, environments SPACE, SCRATCH, EXTEND, RENAME, RELEASE, and QUOTA). If the UCB cannot be found or another environment is active, UNITTYPE is defaulted to 3380.

Note: For SMS-managed allocations, BrightStor CA-Allocate's ability to modify this variable is contingent upon having access to the appropriate system control blocks.

UNITTYPED

Format type = character
Maximum size = 8
Default = '3380'
VSAM Data Component

The UNITTYPED variable contains a string value that specifies the type of unit for space type conversions involving the VSAM data component. A table listing the valid values and device characteristics of each can be found in the UNITTYPE variable description.

UNITTYPEI

Format type = character
Maximum size = 8
Default = '3380'
VSAM Index Component

The UNITTYPEI variable contains a string value that specifies the type of unit for space type conversions involving the VSAM Index Component. A table listing the valid values and device characteristics of each can be found in the UNITTYPE variable description.

USER

Format type = character
Maximum size = 8
Non-VSAM and VSAM

The USER variable is retrieved from the security system if RACF, eTrust CA-TOP SECRET, or CA-ACF2 is active.

Otherwise, it is the TSO user ID in TSO and null, '', in batch.

For ACS, this value is established by the requestor of the processing ACS.

Note: For a DFHSM RECALL or RECOVER, this variable contains the user ID of the issuer of the allocation request. This should always be the user ID associated with the DFHSM address space.

For more information, see the description of variable "HSMUSER" and the section **DFSMShsm Support** in the chapter "**Concepts and Facilities**."

VAMENVIR

Format type = character
Maximum size = 7
Non-VSAM and VSAM

The VAMENVIR variable lets you know why the ASR was entered. Possible values are:

ACS	ACS processing request.
ALLOC	Allocation — called by batch initiator or dynamic allocation (SVC 99).
DEFINE	Catalog management VSAM define request.
EOV	Out of space condition for a non-VSAM data set. An output data set has used all space, exceeded 16 extents, or cannot satisfy its requirements for additional space on a volume and has no additional volumes to use for output.
EOV_VSAM	Out of space condition for a VSAM data set. An output data set has used all space or cannot satisfy its requirements for additional space on a volume and has no additional volumes to use for output.
EXTEND	DADSM extend request.
OLD	Allocation request for an already existing data set.
QREBUILD	The QREBUILD environment is entered one time for each volume that is online during the Quota Table REBUILD operation.
QSCAN	The QSCAN environment is entered one time for each data set, on each volume, that is subject to QUOTA processing during the Quota Table REBUILD operation.
QUOTA	The QUOTA environment is entered after the DEFINE, SPACE, EXTEND, RELEASE, SCRATCH, and RENAME environments, each time they are entered.
RELEASE	DADSM release space request.
RENAME	DADSM rename request.
SCRATCH	DADSM scratch request.
SPACE	DADSM space request.

See the section **Environments** in the chapter "**Concepts and Facilities**" for more information on these environments.

VAMRLSE

Format type = character
Maximum size = 8
Non-VSAM and VSAM

The VAMRLSE variable contains a string value that specifies the release number of the BrightStor CA-Allocate running.

VLCT

Format type = numeric
Non-VSAN and VSAM
Non-SMS-Managed and SMS-Managed

Volume-count for Disk Data sets: For non-VSAM, VLCT initially contains the volume count value as specified for MVS allocations. For example, the JCL statement VOL=(,,,20) indicates that the allocation can use as many as 20 different volume serial numbers. Normally VLCT contains a value of one, but in this example, VLCT contains 20. VLCT can be used as an output variable in the ALLOC environment only. It is often used to reduce the specified volume count to one.

Volume-count for Tape Data sets: if the volume count is omitted or if the specified count is 1 through 5, the MVS maximum number allowed is 5. If the specified count is 6 through 20, the MVS maximum number allowed is 20. If the specified count is greater than 20, the maximum MVS number allowed is a multiple of 15 plus 5, specifying VOL=(,,,20) or any number higher than 5 in the JCL. In the ALLOC Environment, BrightStor CA-Allocate can be used to either increase or decrease the VLCT, to control (within the limitations described in the preceding sentence) how many tapes a data set is allowed to extend upon.

For VSAM, a JCL DEFINE job results in VLCT being non-zero. The only time BrightStor CA-Allocate can modify VLCT for VSAM is in the ALLOC environment.

Note: Setting VLCT to a value outside the range supported by MVS can cause unpredictable results. Currently, supported values are 255 for non-SMS-managed and 59 for SMS-managed.

Note: For SMS-managed setting VLCT to a value greater than the number of volumes in the SMS Storage Group can cause unpredictable results.

Note: Changing the values in both the VLCT and NVOL variables for the same non-SMS-managed new data set allocation (that is, redirected by BrightStor CA-Allocate) can cause unpredictable results.

VOLSER

Format type = character
Maximum size = 8
Non-VSAM and VSAM

The VOLSER variable contains the volume serial number associated with the current allocation request. This variable is available in the EXTEND, QUOTA, QREBUILD, and QSCAN environments:

EXTEND and QUOTA	The volume serial number where the data set operation is occurring.
QREBUILD	The volume serial number being offered for inclusion in the Quota Table rebuild process.
QSCAN	The volume serial number where the data set that is being accumulated is located.

VREFDDN

Format type = character
Maximum size = 1
Non-VSAM

The VREFDDN variable indicates whether or not a volume referback to a prior ddname in the same step was specified for the current allocation request. The values it can contain are:

Y	VOL=REF=*.ddname was specified
N	none was specified

This variable can only be changed in the ALLOC environment. The single possible output value is:

N disable any VOL=REF=*.ddname" specified

When a disk allocation is redirected, any volume referback originally specified for the allocation is automatically disabled. A detailed explanation can be found in "Volume Referbacks / Unit Affinity and Disk-only Allocations" .

Any volume referback is also disabled for TAPE-to-DISK and DISK-to-TAPE device-type conversions, but not for TAPE-to-TAPE allocation changes. A detailed explanation can be found in the section **Device-Type Conversions** in the chapter "**Concepts and Facilities**."

VREFDSN

Format type = character
Maximum size = 1
Non-VSAM

The VREFDSN variable indicates whether a volume referback to a specific data set name was specified for the current allocation request. The values it can contain are:

Y	VOL=REF=*.ddname was specified
N	none was specified

This variable can only be changed in the ALLOC environment. The single possible output value is:

N disable any VOL=REF=*.dsname specified.

When a disk allocation is redirected, any volume referback originally specified for the allocation is automatically disabled. A detailed explanation can be found in “Volume Referbacks / Unit Affinity and Disk-only Allocations.”

Any volume referback is also disabled for TAPE-to-DISK and DISK-to-TAPE device-type conversions, but not for TAPE-to-TAPE allocation changes. A detailed explanation can be found in the section **Device-Type Conversions** in the chapter ”**Concepts and Facilities**.”

VREFSTP

Format type = character
Maximum size = 1
Non-VSAM

The VREFSTP variable indicates whether a volume referback to a prior ddname in a previous step was specified for the current allocation request. The values it can contain are:

Y	VOL=REF=*.stepname.ddname was specified
N	none was specified

This variable can only be changed in the ALLOC environment. The single possible output values:

N	disable any VOL=REF=*.stepname.ddname specified

When a disk allocation is redirected, any volume referback originally specified for the allocation is automatically disabled. A detailed explanation can be found in “Volume Referbacks / Unit Affinity and Disk-only Allocations.”

Any volume referback is also disabled for TAPE-to-DISK and DISK-to-TAPE device-type conversions, but not for TAPE-to-TAPE allocation changes. A detailed explanation can be found in the section **Device-Type Conversions** in the chapter ”**Concepts and Facilities**.”

WEEKDAY

Format type = character
Maximum size = 9
Non-VSAM and VSAM

The WEEKDAY variable contains the day of the week. Possible values for the WEEKDAY variable are:

SUNDAY	MONDAY	TUESDAY	WEDNESDAY
THURSDAY	FRIDAY	SATURDAY	

WRITE_EOFM

Format type = character
Maximum size = 1
Non-VSAM
Non-SMS-Managed

The WRITE_EOFM variable is a "yes" or "no" flag indicating whether or not an End-Of-File Mark should be written for eligible data sets. Only non-SMS-managed, NonVSAM, Physical Sequential data sets are eligible for this support. The default value is 'N'.

This variable can only be changed in the ALLOC Environment. The single possible output value is:

Y Initialize applicable data sets with an end-of-file mark

Note: This support will be limited only to new data set allocations that are completely redirected by BrightStor CA-Allocate. Those that are redirected but deferred, re: PLSOPT9, will not be considered eligible.

Alias : EOFM

XMODE

Format type = character
Maximum size = 7
Non-VSAM and VSAM

The XMODE variable tells you whether the allocation is being requested by a batch job, a started task, or a TSO user. Possible values for the XMODE variable are:

BATCH TSO TASK

Chapter 5

Operation

The MVS console operator interacts with BrightStor CA-Allocate using the MVS START, MODIFY, and STOP commands, but you can test your parameter files (VKGPARMS, VDSPROG, QSCAN, QREBUILD, QCONFIG, and VDSTORGP) without actually installing BrightStor CA-Allocate into your operating system.

When using the QSYNC process of the Quota selectable unit, the BrightStor CA-Allocate Started Task must be left running at all times (RESIDENT=YES). For a detailed description of the important periodic actions that require this residency see the section **The QSYNC Process** in the chapter "**Quota**."

When not using Quota, the Started Task does not need to be left up to enable the interface to control allocations. In this case, the Started Task only controls starting, modifying, and stopping the interface. The installation determines whether to leave the Started Task up (using the RESIDENT= subcommand, explained below).

Testing Parameter Files

Using the TESTASRS member of the Install Library, you can syntax check the PARM files before installing the product.

TESTASRS executes the compiler in standalone batch, instead of from the BrightStor CA-Allocate Started Task. If your site is licensed for Allocation Manager and Quota, then all of the parameter files are required. If your site is only licensed for the Allocation Manager, then only VKGPARMS, VDSPROG,VDSTORGP, and CONFIG are required.

Operator Command Options

Various options are allowed for the START and MODIFY commands.

If the Started Task is not left up and running, use the MVS START (S) command to enter options. If the Started Task is already running, use the MVS MODIFY (F) command to enter options. The syntaxes of the START and MODIFY commands are slightly different, as the following examples illustrate:

```
S VAM,PARM='option1,option2,option'          START command
F VAM,option1,option2,option...              MODIFY command
```

Note: The options are separated by commas, not spaces.

TABLE 5-1 Operator Commands, Options, and Subcommands

CMD	OPTIONS	SUBCOMMANDS
START	{INSTALL |	
	REMOVE |	
	REFRESH |	
	STATUS |	[,ALIASLVL={1 | 2 | 3 | 4}]
		[,ASRSIZE={1 | 2 | 3}]
	PARMREF |	
	DECOMP=}	
		[,{ACTIVE |
		DORMANT}]
		[,ALIASLVL={1 | 2 | 3 | 4}]
		[,ASRSIZE={1 | 2 | 3}]
		[,DIAGS=]
		[,{HSMON |
		HSMOFF}]
		[,{HSMDIAG | or NOHSMDIAG
		[,PROG={VDSPROG | Mbr Name}]
		[,QCONFIG={QCONFIG | Mbr Name}]
		[,QCONFIGSZ=]
		[,RESIDENT={YES | NO}]
		[,STORGP={VDSTORGP | Mbr Name}]
		[,VANTKEY= userid]

CMD	OPTIONS	SUBCOMMANDS
MODIFY	Same as the START command	Same as the START command with the addition of:
		[,ASRSIZE={1|2|3}]
		[,{HSMDIAG| or NOHSMDIAG
		[,{HSMON|
		HSMOFF}]
		[,QREBUILD]
		[,QSYNC]
		[,QRESETWM]
STOP	none	

Notes for Table 5-1 :

1. The operator commands described above, and in more detail below, are not required parameters. They only override what is specified as the default in member VKGPARMS of the partitioned data set pointed to by the PARMS DD statement in the Started Task (from now on referred to as the PARMLIB).
2. For more information about VKGPARMS, see "**Notes for Figure 3-2**", and the section **VKGPARMS Parameters** in the chapter "**Implementation**."
3. If more than one of these options are specified, BrightStor CA-Allocate acts only on the last one.

START Command

The MVS console operator starts BrightStor CA-Allocate with the following command:

```
S VAM,PARM='option1,option2,option3,...option'
```

The default options, defined in the VKGPARMS member of PARMLIB, may be overridden by putting them on the EXEC statement in the VAM cataloged procedure, or by entering them at the console.

Whenever a default option is overridden, the value specified becomes the temporary default and remains so until either the Started Task terminates or another command (that is, REFRESH, PARMREF) is issued. One reason to keep the Started Task running is to maintain the new default values that have been specified.

Caution: If RESIDENT=YES is used, any START command, even one done simply to check status (STATUS), leaves the Started Task running. If a Started Task is already running at the time the START command is entered, then two started tasks are running simultaneously, and both try to respond to future MODIFY or STOP commands.

INSTALL Option

The PLSACTN parameter in the VKGPARMS member of PARMLIB specifies the default action the started task is to take. As delivered, the default action is 'INSTALL', which causes BrightStor CA-Allocate to dynamically install its allocation interfaces into the operating system.

If compiler errors occurred on the first installation attempt and BrightStor CA-Allocate was started 'RESIDENT=YES', neither the REFRESH nor REMOVE commands works because it did not successfully install. Therefore, after correcting the compiler error(s) you must retry installing it using the command:

```
F VAM,INSTALL
```

If the compiler errors persist and you want to remove the Started Task from the system, issue the STOP command as shown:

```
STOP VAM
```

Note: To avoid potential compile problems during the BrightStor CA-Allocate startup, make use of the TESTASRS (documented at the beginning of this chapter) batch compiler.

REMOVE Option

The REMOVE option is used to initiate a normal shutdown. If the Started Task is not running with RESIDENT=YES, enter the following START operator command to uninstall BrightStor CA-Allocate.

```
S VAM,PARM='REMOVE'     START command
```

If the Started Task is invoked with RESIDENT=YES, the following MODIFY operator command should be entered to remove it and to get the Started Task to shut down normally.

```
F VAM,REMOVE            MODIFY command
```

REFRESH Option

The REFRESH option is used to cause the Started Task to load any parameter files that have been changed since BrightStor CA-Allocate was installed. These files are:

1. Storage Group Definitions — PLSSTRMB (VDSTORGP)
2. Allocation Selection Routines — PLSVDSMB (VDSPROG)
3. Quota Configuration File — PLSQCFMB (QCONFIG), PLSQRBMB (QREBUILD), PLSQSCMB (QSCAN)

For example, the REFRESH option can be used with the STORGP= and PROG= options to change the member names of these two parameter files and to put them into effect.

```
F VAM,REFRESH,PROG=MYRULES,STORGP=MYPOOLS
```

Caution: REFRESHes should not be done during high activity periods. If at all possible, choose a time when few allocations, if any, are occurring. If a REFRESH occurs during the volume selection process, unpredictable results may occur.

Note: If you make use of the VAMSPARE feature of BrightStor CA-Vantage Storage Resource Manager, issuing the REFRESH command will cause the removal of the temporary additions of volumes to CA-Allocate STORGRPs. For more information regarding the VAMSPARE function, refer to the *BrightStor CA-Vantage Storage Resource Manager Reference Guide* and the *BrightStor CA-Vantage Storage Resource Manager User Guide.*

STATUS Option

The STATUS option, when specified on the START or MODIFY command, shows BrightStor CA-Allocate's current status on the master console:

```
S VAM,PARM='STATUS'     START command
F VAM,STATUS            MODIFY command
```

If BrightStor CA-Allocate is not installed, the following message is displayed:

```
VAM0011 CA-Allocate IS NOT INSTALLED
```

If BrightStor CA-Allocate is installed, the following messages are displayed:

```
VAM0012 CA-Allocate IS RUNNING WITH PARAMETERS AS FOLLOWS:
VAM0366 STORGRP                       = <VDSTORGP member name>
VAM0366 PROG                          = <VDSPROG ASR name>
VAM0366 QREBUILD                      = <QREBUILD ASR name>
VAM0366 QSCAN                         = <QSCAN ASR name>
VAM0366 QCONFIG                       = <QCONFIG member name>
VAM0366 DISK QUOTA TABLE DSN          = <QT dataset name>
VAM0012 ALIASLVL                      = <1 or 2 or 3 or 4>
VAM0012 ASRSIZE                       = <1 or 2 or 3>
```

PARMREF Option

The PARMREF option instructs the Started Task to get a new copy of the VKGPARMS member of PARMLIB. After this new copy is obtained PARMREF finishes by doing a REFRESH. This option is used when the default value of one of the parameters in VKGPARMS is changed.

For example, suppose the default VDSPROG ASR that is to be used by the Started Task is changed to MYPROG from VDSPROG in VKGPARMS. A 'F VAM,REFRESH' command still compiles the VDSPROG ASR because the Started Task's copy of VKGPARMS still indicates VDSPROG. However, a 'F VAM,PARMREF' command first retrieves a new copy of VKGPARMS, and then proceeds with a 'REFRESH', which results in the MYPROG ASR being compiled.

Note: If any parameters are removed from (or commented out) in VKGPARMS, they will not change back to their default values. For example, in the above scenario, if PLSVDSMB(MYPROG) is removed from VKGPARMS and a PARMREF is issued, the MYPROG ASR will still be the active ASR and will be compiled again; the default VDSPROG will not.

DECOMP= Option

The DECOMP option is used to create a “snapshot” report of the ASRs, Pools, and Quota Configuration File. This decompiled diagnostic report shows the way BrightStor CA-Allocate was set up the last time it was INSTALLED or REFRESHED. The report is dynamically allocated to a SYSPRT file in the Started Task DD name 'VAMPTR'.

The DECOMP option, when specified on the START or MODIFY command, causes BrightStor CA-Allocate to display its current status on the operator's console:

```
S VAM,PARM='DECOMP'     START command
F VAM,DECOMP            MODIFY command
```

ACTIVE Subcommand

The ACTIVE subcommand directs the Started Task to set BrightStor CA-Allocate to the active state. It overrides what was specified in the PLSMODE parameter in the VKGPARMS of PARMLIB. As delivered, the default value is 'ACTIVE'.

This is the normal mode of operation.

ALIASLVL= Subcommand

The ALIASLVL= subcommand is needed only in installations making use of the MVS/ESA Multi-Level Alias (MLA) facility. The default value is what was specified in the PLSALLVL parameter in the VKGPARMS member of PARMLIB. As delivered, the default value is '1'.

This facility allows different user catalogs to be pointed to using 1, 2, 3, or 4 levels of data set name qualifiers. If an installation uses MLA and has it set to use more than one level of qualification, this subcommand must be used to make sure BrightStor CA-Allocate selects the correct catalog whenever it needs to do so.

ASRSIZE= Subcommand

The ASRSIZE is only needed in installations that have exceedingly large or complex Allocation Selection Routines that cannot be successfully compiled because of a “VDSCOMP STACK OVERFLOW” or “SYNTAX ERROR: PROGRAM TOO LARGE” error. The default value is 1, which allocates the historical 256KB for the compiled length of the ASR(s). Additional values of 2 or 3 can double or triple the amount to 512 or 1024 KB, respectively.

Note: Increasing the ASRSIZE beyond the default value can require an increase of the REGION specified by for the BrightStor CA-Allocate Started Task.

DIAGS= Subcommand

The DIAGS= subcommand specifies the jobname of a job that receives diagnostic messages. The default value is what was specified in the PLSDIAGS parameter in the VKGPARMS member of PARMLIB. As delivered, the default value is ' ' (no jobname).

This subcommand should be used in cases were a VDSDIAGS DD cannot easily be put into a job step (like DB2, or DFHSM). Specifying DIAGS= with no jobname is the default, and means that no jobname produces diagnostic messages. After a DIAGS=jobname is specified, diagnostic messages continues to be produced for the job, until the DIAGS= with no jobname is specified.

DORMANT Subcommand

The DORMANT subcommand directs the Started Task to set BrightStor CA-Allocate to the dormant state. It overrides what was specified in the PLSMODE parameter in the VKGPARMS member of PARMLIB. As delivered, the default value is 'ACTIVE'.

When running in DORMANT mode, a job or TSO session is only processed when a special DD name is present. The DD name is specified in the PLSDRMNT parameter in the VKGPARMS member of PARMLIB. As delivered, the DD name is '#VAMTST#'. This subcommand is typically used for testing.

HSMDIAG Subcommand

The HSMDIAG subcommand instructs the BrightStor CA-Allocate Started Task to activate diagnostics in the DFSMShsm address space. The diagnostics remain active until the NOHSMDIAG subcommand is issued.

Caution: Due to substantial spool output, these diagnostics should only be activated in special cases.

NOHSMDIAG Subcommand

The NOHSMDIAG subcommand deactivates the diagnostics activated by HSMDIAG. As delivered, the default value is ‘NOHSMDIAG.’

HSMON Subcommand

The HSMON subcommand instructs the BrightStor CA-Allocate Started Task to install the SMS Subsystem Interface hook during an INSTALL operation and/or to activate the DFSMShsm RECALL/RECOVER redirection feature.

For details, see the sections **DFSMShsm During RECALL/RECOVER** and **ASR Variables for HSM RECALL** and RECOVER Operations in the chapter "**Concepts and Facilities**."

HSMOFF Subcommand

The HSMOFF subcommand deinstalls the SMS Subsystem Interface hook installed by HSMON. The default value is 'HSMOFF.'

PROG= Subcommand

The PROG= subcommand indicates which member in the library pointed to by the VDSPROG DD statement to compile and use as the Allocation Selection Routine. The default value is what was specified in the PLSVDSMB parameter in the VKGPARMS member of PARMLIB. As delivered, the default value is 'VDSPROG'.

QCONFIG= Subcommand

The QCONFIG= subcommand specifies the member name that contains the Quota Configuration File. The default value is what was specified in the PLSQCFMB parameter in the VKGPARMS member of PARMLIB. As delivered, the default value is 'QCONFIG'.

Note: The statements contained in this member control the actions of the Quota selectable unit. For more information see the section **Quota Configuration File** in the chapter "**Quota**."

Note: The default value for the QSCAN ASR is specified in the PLSQSCMB parameter in the VKGPARMS member or PARMLIB. The QSCAN OPTION of the Quota Configuration File overrides the default name of 'QSCAN'.

Note: The default value for the QREBUILD ASR is specified in the PLSQRBMB parameter in the VKGPARMS member of PARMLIB. The QREBUILD OPTION of the Quota Configuration File overrides the default name of 'QREBUILD'.

QCONFIGSZ= Subcommand

The QCONFIGSZ= subcommand specifies how many 80 byte lines to provide for when getting storage for the Quota Configuration File, if you need substantially more than the default of 3000. Because lines are generally compacted to less than 80 bytes, the storage acquired usually supports more lines than you specify. You can specify up to 99999.

RESIDENT= Subcommand

The RESIDENT= subcommand tells the Started Task whether to wait for an operator command or to terminate.

The default value is what was specified in the PLSRES parameter of PARMLIB. As delivered, the default value is 'YES'.

'RESIDENT=YES' instructs the Started Task to wait for a command. Specifying 'RESIDENT=NO' instructs the task to terminate after it has completed the current command. If it is necessary to uninstall after 'RESIDENT=NO', the START command must be issued again with the REMOVE option. For example:

```
S VAM,PARM='REMOVE'
```

Note: 'RESIDENT=YES' is required if the Quota selectable unit is being used.

STORGP= Subcommand

The STORGP= subcommand indicates which member in the library pointed to by PLSPRGDS VKGPARM contains the storage group definitions to be used. The default value is what was specified in the PLSSTRMB parameter in the VKGPARMS member of PARMLIB. As delivered, the default value is 'VDSTORGP'.

VANTKEY= Subcommand

If your site is not licensed for BrightStor CA-Vantage, this option is meaningless. VANTKEY specifies the TSO userid of a BrightStor CA-Vantage user to which the BrightStor CA-Allocate Started Task sends all operation type messages. For example, the commands to use to send messages to BrightStor CA-Vantage user 'ISPKCH1' are, depending on the residency of the Started Task:

```
S VAM,VANTKEY='ISPKCH1'      START command
F VAM,VANTKEY=ISPKCH1        MODIFY command
```

If the user is not logged on then the message is lost. If the user is logged onto BrightStor CA-Vantage then a special window pops up displaying all of these messages.

MODIFY Command

The MODIFY command is used to change the configuration of the Started Task that remained active after it was started and installed with RESIDENT=YES. As shown in Table 5-1, “Operator Commands, Options, and Subcommands”, the parameters available for the MODIFY command are identical to the ones for the START command.

The MODIFY command can be used to change the current Allocation Selection Routine to another one that also resides in the data set pointed to by the PLSPRGDS parameter in the VKGPARMS member of PARMLIB. For example, the command to switch to the ASR in member NEWPROG would be:

```
F VAM,REFRESH,PROG=NEWPROG
```

The MODIFY command can be used to generate VDSDIAGS Trace Data for the ACS Environment, or for any job or started task without the need for the VDSDIAGS DD statement. For example, the commands to “turn on”, and then “turn off” VDSDIAGS Tracing for the BrightStor CA-Disk Backup and Restore DMSAR (Auto Restore) Started Task is:

```
F VAM,REFRESH,DIAGS=DMSAR            "on"
F VAM,REFRESH,DIAGS=                 "off"
```

The MODIFY commands to use change this product's operating mode from “active” to “dormant”, and from “dormant” to “active” are:

```
F VAM,ACTIVE
F VAM,DORMANT
```

The MODIFY command is the recommended way to remove this product because, unlike the STOP command, MODIFY does not terminate the Started Task if the Started Task fails to uninstall itself. The command to initiate the shutdown is:

```
F VAM,REMOVE
```

QREBUILD Subcommand

The QREBUILD subcommand directs the Started Task to initiate a Quota Table rebuild operation. It is applicable only to the Quota selectable unit. For more information, see the section **The QREBUILD Process** in the chapter ”**Quota**.”

QSYNC Subcommand

The QSYNC subcommand directs the Started Task to synchronize the Disk and CSA Quota Tables. It is applicable only to the Quota selectable unit. For more information, see the section **The QSYNC Process** in the chapter “**Quota**.”

QRESETWM Subcommand

The QRESETWM subcommand directs the Started Task to initialize the water marks to the current allocation amount. This process is followed by a QSYNC operation which re-synchronizes the CQT with the DQT. It is applicable only to the Quota selectable unit.

For more information, see the section **QRESET Statement** in the chapter "**Quota**."

STOP Command

The STOP command can be used to uninstall the BrightStor CA-Allocate interface:

```
P VAM
```

After the STOP command is issued, the Started Task removes the allocation hooks and terminates (after a short wait). This delay is intentional; it is meant to permit allocations in process to finish using the storage that BrightStor CA-Allocate frees.

Note: The STOP command is NOT the recommended method of removing BrightStor CA-Allocate from the system. Instead do a 'MODIFY REMOVE' as follows:

```
F VAM,REMOVE
```

The only difference between the two commands is that with the STOP command, even if an error occurs, the started task stops executing. With 'MODIFY REMOVE' the started task does not stop if an error occurs.

After successfully uninstalling itself, BrightStor CA-Allocate issues the following message to the SYSLOG:

```
VAM000I CA-Allocate SHUTDOWN COMPLETED
```

Chapter

6

User Exits

Up to ten program routines or User Exits can be supplied to which BrightStor CA-Allocate can give control. The ASR CALL statement can invoke these user-written routines from any point in the ASR.

The exit can return data to the ASR by storing the information into the user variables, N0 to N40 and C0 to C40. The ASR can then use the new contents of the user variables after the CALL statement. Character variables are null terminated (X'00'). A flag byte precedes each user variable. Do not change it.

All exit routines must be re-entrant and the exits must be named VDSEXIT0, 1, 2 ... through VDSEXIT9. Because the User Exits gain control during the allocation process, functions like OPEN, CLOSE, PARTREL, and dynamic allocation are not supported. Abends and other unpredictable results are likely.

Table 6-1 contains a list of sample user exits distributed in the SAMSAMP File on the installation tape.

TABLE 6-1 Available User Exits

Name	Description
EXIT01	Enhanced version of EXIT05.
EXIT02	Sends a message to a specific “logged on” TSO User ID.
EXIT03	Retrieves the 8-character ACF2 Logon ID (User ID).
EXIT04	Retrieve &CATDSN for non-VSAM allocations.
EXIT05	Reports which STORGRP a specific volume is within.
EXIT06	Checks to see if a specific data set is cataloged (pre-DFP Release 3 systems only) and returns the VOLSER.
EXIT07	Allow concatenation of character strings.
EXIT08	Performs a DSN-level security check.
EXIT09	Writes SMF records. Default record ID is 213.
EXIT10	Determines the CYL amount of a VSAM records allocation.
EXIT13	Verifies that &C0 contains a valid UNIT.

Name	Description
EXIT14	Issues a WTOR from an ASR.

Installing User Exits with SMP/E

Use member SMPEXIT in the Install Library. After this job has executed, your User Exits are:

1. Incorporated into the BrightStor CA-Allocate loadlib

2. Accessible at the next start-up of BrightStor CA-Allocate

Note: A REFRESH does not load User Exits into CSA! REFRESH only implements ASR and pool definition changes. Loadlib changes require that BrightStor CA-Allocate be “recycled” (brought down, and then back up) in order for them to take effect.

Assembling and Linking User Exits

Below is the JCL to assemble and link the sample user exits.

```
//ASMHCL         EXEC ASMHCL,PARM.ASM='NODECK,OBJECT,RENT',
//    PARM.LKED='LIST,LET,XREF,MAP'
//*
//ASM.SYSLIB     DD DISP=SHR,DSN=SYS1.MACLIB
//                  DD DISP=SHR,DSN=VAM.INSTALL
//ASM.SYSIN      DD DISP=SHR,DSN=VAM.INSTALL(YOUR EXIT)
//LKED.SYSLMOD   DD DISP=SHR,DSN=VAM.LOADLIB
//LKED.SYSIN     DD *
   INCLUDE SYSLMOD(VDSINTRP)
   MODE AMODE(31),RMODE(ANY)
   ORDER GBLVDS
   ENTRY GBLVDS
   SETCODE AC(1)
  NAME VDSINTRP(R)
//
```

Chapter 7

Quota

Storage Administration is complicated by users who overuse their allocated disk space. These are most often developers and ad-hoc system users who do not have specific disk space needs and are allowed to allocate space from a shared disk pool. Since multiple users share the same pool, they must be good neighbors to assure everyone enough space.

Unfortunately, cooperation in sharing disk pools does not work reliably. Users still use more space than planned and deplete the shared pool. Then other users of the pool demand a solution from the Storage Administrator.

With the introduction of SMS, running out of shared pool space grew more serious because SMS storage groups are much larger and many more users share the same pool.

Quota resolves this disk sharing problem by monitoring pool usage by user, group, or department and by giving Storage Administrators absolute control over how much total at any level can allocate. Quota is not usually implemented for every user of storage space or for every application. Quota is intended for users and applications that require space monitoring.

How Quota Works

Quota can dynamically capture and track disk allocation statistics for individuals, groups of users, or groups of data. Such groups of users or data are called Quota Groups. Quota can also set and optionally enforce limits on the amount of space allocated to Quota Groups at any one time to ensure equitable access to disk space in common storage pools. These limits can be assigned in units of kilobytes, megabytes, tracks, or cylinders. The statistics are captured when the system allocates, extends, renames, deletes, or releases idle space.

Quota tracks allocation statistics in three ways. Initial usage statistics are collected when Quota is brought up the first time. The impact a proposed change has on a group's allocation statistics is determined before the change is allowed. The actual allocation amount of the change is collected at the time it is made. Quota keeps the allocation statistics in real computer memory and periodically copies them to a disk file.

Initial allocation statistics are collected by a subtask that scans online disk volumes. One parameter file determines which volumes are tracked. Another parameter file analyzes each data set on the selected volumes to determine which group is accountable for which data set's allocation amount.

Quota uses two interfaces in the operating system to keep track of proposed and actual changes to disk allocation statistics. The first interface is just before the allocation change takes place. A subroutine determines the impact that the proposed change has on a group's allocation statistics. This is the point of control for how much space is used by any group. This is where Quota can be directed to disallow requests that let a group exceed its predefined limit.

Quota's second interface in the operating system is after the allocation change has occurred. At this point, Quota updates the applicable group's allocation statistics. The memory copy of the allocation statistics is updated in real-time. The disk copy is updated periodically at whatever interval was specified in the configuration parameter file.

Quota has three implementation modes:

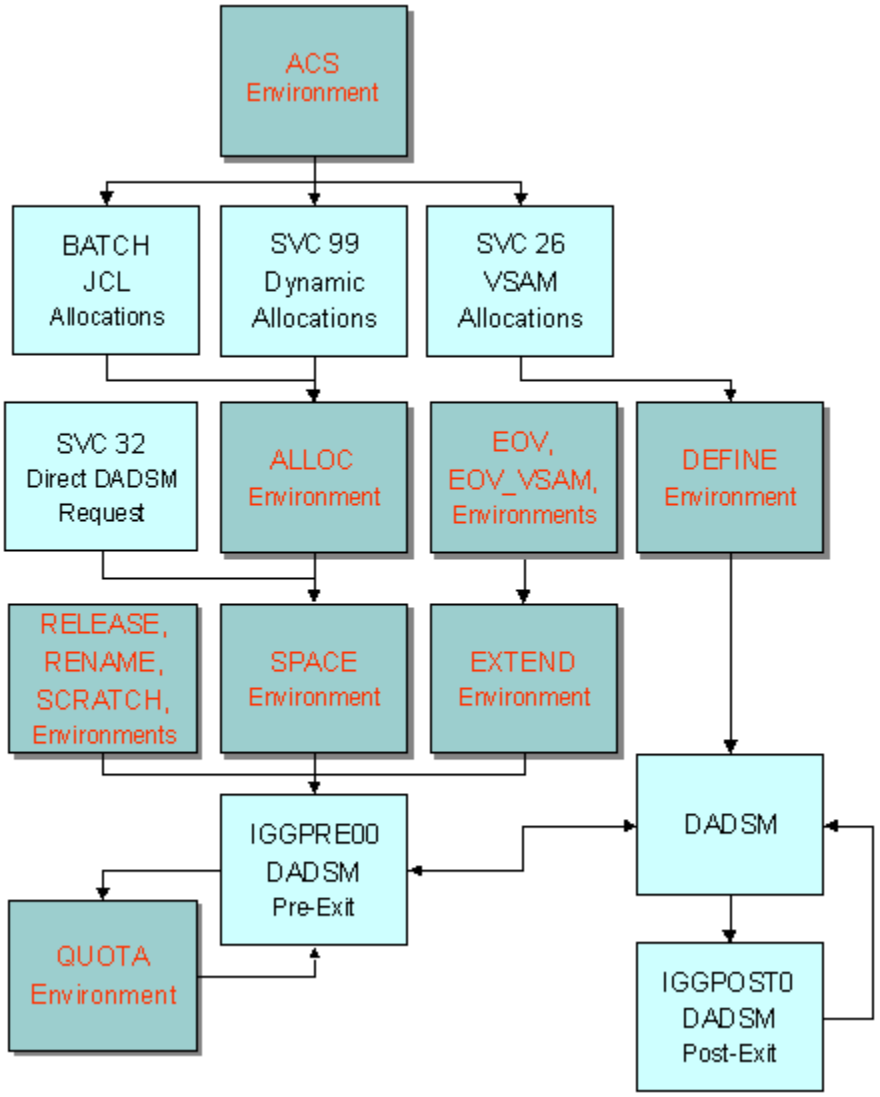
- Silent mode monitors allocations and informs the storage administrator when quota limits are exceeded.
- Warning mode warns users when they exceed their quota limits.
- Fail mode disallows allocations that allow a group to exceed its predefined limit.

Note: Running with PLSKBYTE (1000) instead of the default (1024) will introduce a loss of precision resulting in the quota allocation amounts overstating what will actually be allocated. If running in fail mode, this may result in allocations being erroneously denied because the quota limit has been, or would otherwise be, exceeded.

System Overview

Quota tracks all DADSM Activity (CREATE, EXTEND, RELEASE, RENAME, SCRATCH) for all data sets, both VSAM and non-VSAM, including SMS-managed. It does this by installing IGGPRE00 (DADSM pre-allocation exit) and IGGPOST0 (DADSM post-allocation exit) at the standard DADSM exit points provided by the operating system. Figure 7-1 shows a simplified view of the relationship between these hooks, BrightStor CA-Allocate other hooks, and the entire allocation process.

FIGURE 7-1 How Quota Tracks DADSM Activities



As Figure 7-1 shows, the QUOTA Environment is entered for every DADSM data set operation for all types of data sets. It is called before the DADSM operation takes place from BrightStor CA-Allocate hook at IGGPRE00. This is the point of absolute control over how much space is used by any group of users. This is where an allocation can be denied to keep a group within its predefined limit. Although there is no Environment at the IGGPOST0 hook point, a significant amount of Quota processing occurs there. Quota first sees whether the DADSM operation that just completed was successful. If it was, Quota updates the memory copy of the quota group's allocation statistics. The memory copy of these statistics is called the CSA Quota Table (CQT). The disk copy is called the Disk Quota Table (DQT).

Table 7-1 shows the sequence of ASR Environments entered during the life cycle of a data set, from the time it is originally allocated, through open, close, extension, rename, and finally deletion.

TABLE 7-1 Life cycle of a data set with Quota

OPERATION	non-VSAM	VSAM
Allocate a new single-volume data set.	ACS	ACS
	ALLOC	DEFINE (cluster)
	SPACE	QUOTA (data component primary)
	QUOTA (primary)	QUOTA (index component primary)
Write to the data set so that an extent is required.	EXTEND	EXTEND
	QUOTA (secondary)	QUOTA (data component secondary)
		QUOTA (index component secondary)
Release unused space in the data set when it is closed.	RELEASE	not applicable
	QUOTA	
Rename the data set.	OLD	QUOTA (data component)
	RENAME	QUOTA (index component)
Write to the data set so that an additional volume is required.	EOV	EOV_VSAM (data component)
	QUOTA (secondary)	QUOTA (data component primary)
		EOV_VSAM (index component)
		QUOTA (index component primary)
Scratch the data set.	SCRATCH (volume 1)	
	QUOTA (volume 1)	QUOTA (data component volume 1)
		QUOTA (index component volume 1)
	SCRATCH (volume 2)	
	QUOTA (volume 2)	QUOTA (data component volume 2)
		QUOTA (index component volume 2)

Variables

TABLE 7-2 QUOTA Related Variables

Variable	QREBUILD	QSCAN	QUOTA	Variable	QREBUILD	QSCAN	QUOTA
ACCT_JOB			I	**PRIMARY**			I
ACCT_STEP			I	**PROCSTEP**			I
APPLIC			I	QHIWATERMARK			I
BLKSIZE		I	I	QHIWATERMARKPERCENT			I
CONTIG			I	QHIWATERMARKTIME			I
C0 to C9 and **C10 to C40**	I/O	I/O	I/O	QLIMITIFNEW		I/O	I/O
DATACLAS		I	I	QLOWATERMARK			I
DMSGROUP			I	QLOWATERMARKPERCENT			I
DMSUSR			I	QLOWATERMARKTIME			I
DSN		I	I	QPARENTIFNEW		I/O	I/O
DSORG		I	I	QUOTAALLOCS			I
DSTYPE		I	I	QUOTACC			I
GROUP			I	QUOTACHG			I
HLQ		I	I	QUOTAFUNC			I
HSMDSN			I	QUOTAGROUP		I/O	I/O
HSMGROUP			I	QUOTALIMITS			I
HSMHLQ			I	QUOTANAMES			I
HSMJOB			I	QUOTAPERCENTS			I
HSMLLQ			I	QUOTASTATS			I
HSMNQUAL			I	QUOTAWPERCENTS			I
HSMUSER			I	QWARNINGIFNEW		I/O	I/O
IFALREADYCAT		I	I	**RECFM**			I
IFALREADYCATVOL		I	I	**SECONDARY**			I
JOB			I	**SPACTYPE**			I
JOBCLASS			I	**STEPNAME**			I

Variable	QREBUILD	QSCAN	QUOTA	Variable	QREBUILD	QSCAN	QUOTA
MGMTCLAS		I	I	STORCLAS		I	I
LLQ		I	I	STORGRP	I	I	I
MODULE			I	SYSID	I	I	I
NEWNAME			I	TIME	I	I	I
NEWNAMEHLQ			I	TODAY	I	I	I
NEWNAMELLQ			I	USER			I
NEWNAMENQUAL			I	VAMENVIR			I
NEWQUOTAGROUP			I/O	VAMRLSE	I	I	I
NQUAL		I	I	VOLSER	I	I	I
N0 to N40	I/O	I/O	I/O	WEEKDAY	I	I	I
PGM			I	CAT_VOL_COUNT	I	I	I
PGMRNAME			I				

Description of ASR Variables

NEWQUOTAGROUP

Format type = character
Maximum size = 50
Non-VSAM and VSAM

The NEWQUOTAGROUP variable is set in the QUOTA ASR environment to the name of the quota group for the new data set name during a rename operation. It communicates the new quota group to the QUOTACHK internal routine.

Alias: NQGRP

QHIWATERMARK

Format type = numeric(sub)
Non-VSAM and VSAM

After the QUOTACHK internal routine is called, the elements of the subscripted variable QHIWATERMARK contain the current quota group high water mark amounts in kilobytes (after the current DADSM operation has completed). Element 1 contains the current high water mark amount for the bottom level quota group in the hierarchy. Element 2 contains the current high water mark amount for the next higher quota group in the hierarchy, and so on up to a maximum of ten levels.

Note: For a rename operation, these values correspond to the quota groups for the new data set name, not the old one.

Alias: QHWM

QHIWATERMARKPERCENT

Format type = numeric(sub)
Non-VSAM and VSAM

After the QUOTACHK internal routine is called, the elements of the subscripted variable QHIWATERMARKPERCENT contain the current quota group high water mark amounts in kilobytes (after the current DADSM operation has completed) as a percent of the quota limit. Element 1 contains the percent for the bottom level quota group in the hierarchy. Element 2 contains the percent for the next higher quota group in the hierarchy, and so on up to a maximum of ten levels.

Note: For a rename operation, these values correspond to the quota groups for the new data set name, not the old one.

Alias: QHWMP

QHIWATERMARKTIME **Format type = character(sub)**
Maximum size = 17
Non-VSAM and VSAM

After the QUOTACHK internal routine is called, the elements of the subscripted variable QHIWATERMARKTIME contain the date and time the current quota group high water mark amounts were reached (after the current DADSM operation has completed). The date and time is in the format mm/dd/yyyy hh:mm:ss. Element 1 contains the date and time for the bottom level quota group in the hierarchy. Element 2 contains the date and time for the next higher quota group in the hierarchy, and so on up to a maximum of ten levels.

Note: For a rename operation, these values correspond to the quota groups for the new data set name, not the old one.

Alias: QHWMT

QLIMITIFNEW **Format type = numeric**
Non-VSAM and VSAM

The QLIMITIFNEW variable is set to a value in the QUOTA environment or QSCAN ASR to specify a quota limit in kilobytes for a new quota group.

Alias: QLIMNU

QLOWATERMARK **Format type = numeric(sub)**
Non-VSAM and VSAM

After the QUOTACHK internal routine is called, the elements of the subscripted variable QLOWATERMARK contain the current quota group low water mark amounts in kilobytes (after the current DADSM operation has completed). Element 1 contains the current low water mark amount for the bottom level quota group in the hierarchy. Element 2 contains the current low water mark amount for the next higher quota group in the hierarchy, and so on up to a maximum of ten levels.

Note: For a rename operation, these values correspond to the quota groups for the new data set name, not the old one.

Alias: QLWM

QLOWATERMARKPERCENT

Format type = numeric(sub)
Non-VSAM and VSAM

After the QUOTACHK internal routine is called, the elements of the subscripted variable QLOWATERMARKPERCENT contain the current quota group low water mark amounts in kilobytes (after the current DADSM operation has completed) as a percent of the quota limit. Element 1 contains the percent for the bottom level quota group in the hierarchy. Element 2 contains the percent for the next higher quota group in the hierarchy, and so on up to a maximum of ten levels.

Note: For a rename operation, these values correspond to the quota groups for the new data set name, not the old one.

Alias: QLWMP

QLOWATERMARKTIME

Format type = character(sub)
Maximum size = 17
Non-VSAM and VSAM

After the QUOTACHK internal routine is called, the elements of the subscripted variable QLOWATERMARKTIME contain the date and time the current quota group low water mark amounts were reached (after the current DADSM operation has completed). The date and time is in the format mm/dd/yyyy hh:mm:ss. Element 1 contains the date and time for the bottom level quota group in the hierarchy. Element 2 contains the date and time for the next higher quota group in the hierarchy, and so on up to a maximum of ten levels.

Note: For a rename operation, these values correspond to the quota groups for the new data set name, not the old one.

Alias: QLWMT

QPARENTIFNEW

Format type = character
Maximum size = 50
Non-VSAM and VSAM

The QPARENTIFNEW variable is set to a value in the QUOTA environment or QSCAN ASR in order to specify the parent of a new quota group.

Note: If the quota group specified does not exist, then the default Quota Group specified in the OPTIONS statement in the QCONFIG parameter file is used as the parent of the new quota group.

Alias: QPIFNU

QUOTAALLOCS

Format type = numeric(sub)
Non-VSAM and VSAM

After the QUOTACHK internal routine is called, the elements of the subscripted variable QUOTAALLOCS contain the current quota group allocation amounts in kilobytes (after the current DADSM operation has completed). Element 1 contains the current allocation amount for the bottom level quota group in the hierarchy. Element 2 contains the current allocation amount for the next higher quota group in the hierarchy, and so on up to a maximum of ten levels.

Note: For a rename operation, these values correspond to the quota groups for the new data set name, not the old one.

Alias: QALLOC

QUOTACC

Format type = numeric
Non-VSAM and VSAM

The QUOTACC variable contains the condition code returned by the QUOTACHK internal routine as follows:

CODE	MEANING
0	All quota groups are below the warning level.
4	One or more quota groups are above the warning level but below the quota limit.
8	One or more quota groups are above the quota limit.

Alias: QACC

QUOTACHG

Format type = numeric
Non-VSAM and VSAM

After the QUOTACHK internal routine is called, the QUOTACHG variable contains the total bytes being allocated or released. This is true regardless of the type of DADSM operation taking place: ALLOC, EXTEND, RELEASE, SCRATCH, or RENAME, as indicated by the QUOTAFUNC variable.

Alias: QCHG

QUOTAFUNC

Format type = numeric
Maximum size = 8
Non-VSAM and VSAM

The QUOTAFUNC variable contains the type of DADSM operation: ALLOC, EXTEND, RELEASE, SCRATCH, or RENAME.

Alias: QFUNC

QUOTAGROUP

Format type = character
Maximum size = 50
Non-VSAM and VSAM

The QUOTAGROUP variable is set in the QUOTA environment to associate an allocation request with a quota group. This variable is used to communicate the quota group to the QUOTACHK internal routine.

Alias: QGRP

QUOTALIMITS

Format type = numeric(sub)
Non-VSAM and VSAM

After the QUOTACHK internal routine is called, the elements of the subscripted variable QUOTALIMITS contain the current quota group allocation limits in kilobytes. Element 1 contains the current allocation limit for the bottom level quota group in the hierarchy. Element 2 contains the current allocation limit for the next higher quota group in the hierarchy, and so on up to a maximum of ten levels.

Note: For a rename operation, these values correspond to the quota groups for the new data set name, not the old one.

Alias: QLIM

QUOTANAMES

Format type = character(sub)
Maximum size = 500 (10 x 50)
Non-VSAM and VSAM

After the QUOTACHK internal routine is called, the elements of the subscripted variable QUOTANAMES contain the current quota group names. Element 1 contains the quota group name of the bottom level quota group in the hierarchy. Element 2 contains the quota group name for the next higher quota group in the hierarchy, and so on up to a maximum of ten levels.

Note: For a rename operation, these are the quota group names for the new data set name, not the old one.

Alias: QNAMES.Quota

QUOTAPERCENTS

Format type = numeric(sub)
Non-VSAM and VSAM

After the QUOTACHK internal routine is called, the elements of the subscripted variable QUOTAPERCENTS contain the allocation amounts (after the current DADSM operation has completed) as a percent of the quota limit. Element 1 contains the percent for the bottom level quota group in the hierarchy. Element 2 contains the percent for the next higher quota group in the hierarchy, and so on up to a maximum of ten levels.

Note: For a rename operation, these values correspond to the quota groups for the new data set name, not the old one.

Alias: QPCT

QUOTASTATS

Format type = numeric(sub)
Non-VSAM and VSAM

After the QUOTACHK internal routine is called, the elements of the subscripted variable QUOTASTATS contain condition codes for each quota group in the hierarchy (after the current DADSM operation has completed). Element 1 contains the condition code for the bottom level quota group in the hierarchy. Element 2 contains the condition code for the next higher quota group in the hierarchy, and so on up to a maximum of ten levels. Possible values are:

CODE	MEANING
0	All quota groups are below the warning level.
4	One or more quota groups are above the warning level but below the quota limit.
8	The quota group is above the quota limit.

Note: For a rename operation, these values correspond to the quota groups for the new data set name, not the old one.

Alias: QSTAT

QUOTAWPERCENTS

Format type = numeric(sub)
Non-VSAM and VSAM

After the QUOTACHK internal routine is called, the elements of the subscripted variable QUOTAWPERCENTS contain the quota group warning percentages. Element 1 contains the warning percent for the bottom level quota group in the hierarchy. Element 2 contains the warning percent for the next higher quota group in the hierarchy, and so on up to a maximum of ten levels.

Note: For a rename operation, these values correspond to the quota groups for the new data set name, not the old one.

Alias: QWPCT

QWARNINGIFNEW **Format type = numeric**
Non-VSAM and VSAM

The QWARNINGIFNEW variable is set to a value in the QUOTA or QSCAN ASR environments in order to specify a quota warning percent (up to 99%) for the new quota group specified in &QUOTAGROUP.

Alias: QWIFNU

The Importance of QUOTAFUNC

BrightStor CA-Allocate has over two hundred ASR variables, which are described in detail in “Description of ASR Variables.” Several of these are Quota-only variables. One warrants special mention: QUOTAFUNC. Quota tracks all DADSM activities. The QUOTAFUNC variable identifies the DADSM function that called the QUOTA Environment: ALLOC, EXTEND, RELEASE, RENAME, or SCRATCH.

ASR writers need to be aware of the DADSM function / activity / operation that called the QUOTA Environment, particularly if they are coding rules to deny allocations to those groups that have exceeded their quota limits.

Denying RELEASE or SCRATCH requests prevents such groups from reducing their allocated disk space so that they are again within their quota limits.

RENAME requests may reduce one group's allocation statistics and increase another's. When tracking RENAME activities, the NEWNAME variable is the one to query and the NEWQUOTAGROUP is the one to “set,” not the DSN and QUOTAGROUP variables. Three other ASR variables are unique to RENAME operations—NEWNAMEHLQ, NEWNAMELLQ, NEWNAMENQUAL—and two others may apply to them—QLIMITIFNEW, QPARENTIFNEW.

The QREBUILD Process

QREBUILD is the process by which the initial disk allocation statistics are collected when Quota is brought up the first time. During a QREBUILD, the VTOCs of all available volumes subject to Quota processing are scanned and the allocation statistics are accumulated.

Note: Non-shared DASD is not available to all systems, so if a QREBUILD is performed in a system that does not have access to all volumes, all information relating to the unavailable volumes is lost. If QUOTA is running on multiple systems with non-shared DASD, insure that QREBUILD is done only on a system having access to ALL the volumes you want to track.

The Quota Limit, Parent Quota Group and Quota Warning Percent are initially specified when the Quota Group is created. Quota Groups can be created four different ways:

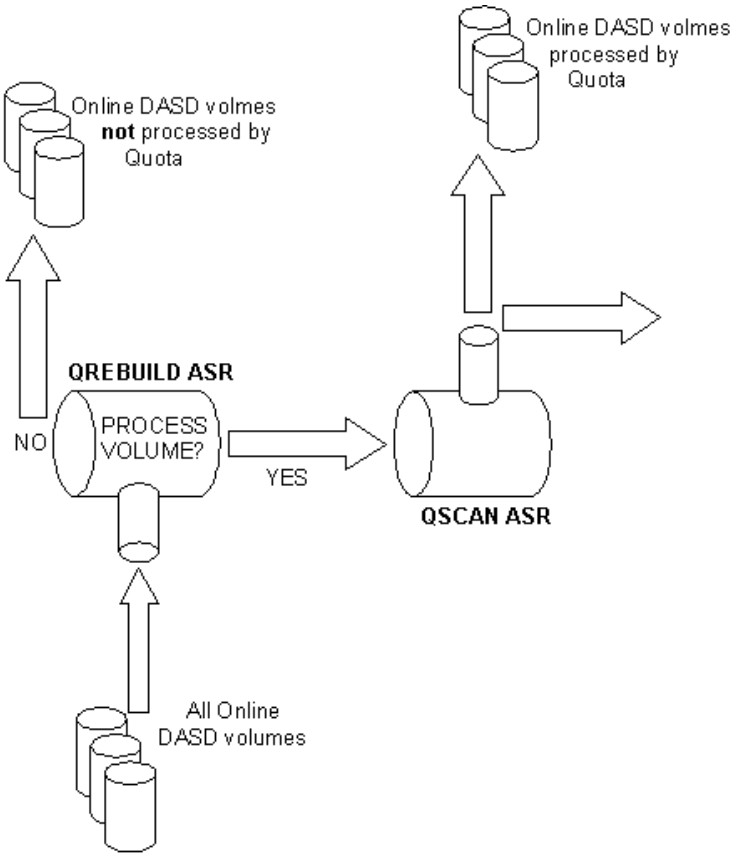
1. Explicitly defining attributes in the QCONFIG file.
2. Dynamically defining attributes in the QSCAN environment.
3. Dynamically defining attributes in the QUOTA ASR Environment when DASD allocation statistics are being tracked.
4. Use the VAMSRV10 API from a batch job or APF-authorized REXX program.

To change attributes to existing Quota Groups, explicit changes need to be defined in the Quota Group in a QUOTADEF statement in the QCONFIG file prior to initiating a QREBUILD operation or use the VAMSRV10 API through a batch job or APF-authorized REXX program.

Note: The QREBUILD operation updates existing Quota Groups for the Quota Limit, Pa rent Quota Group and the Quota Warning Percent. The VAMSRV10 API is used only for changes in the Quota Limit and Quota Warning Percent.

If ASRQUOTAGROUPS(YES) was specified on the OPTIONS statement in the Quota Configuration File, new Quota Groups are automatically generated as the QREBUILD process progresses. Figure 7-2 shows a simplified view of this process.

FIGURE 7-2 QREBUILD Process



As Figure 7-2 illustrates, the QREBUILD Allocation Selection Routine (ASR) is the parameter file that determines which volumes are tracked. The QSCAN ASR determines which group is accountable for which data set's allocation amount. Only the memory copy of the statistics (the CQT) is updated. After the QREBUILD process is completed, a QSYNC (described below) is started to copy the allocation statistics from the CQT to the disk copy, the Disk Quota Table (DQT).

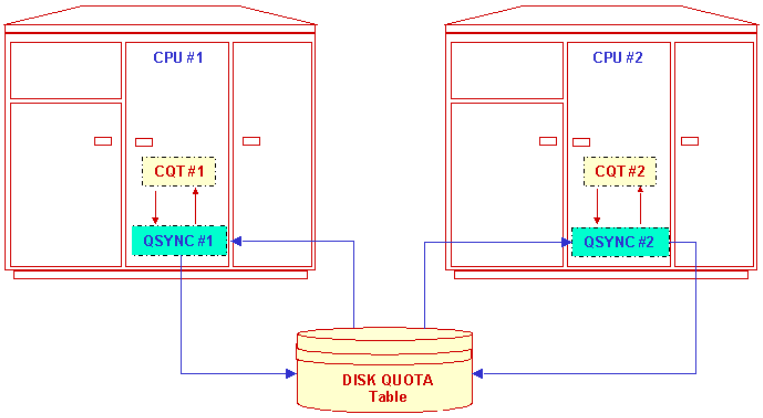
A QREBUILD operation is automatically started whenever a QRESET or QUOTADEF statement is found in the Quota Configuration File during BrightStor CA-Allocate start-up or during a REFRESH operation. It may also be requested on demand by specifying QREBUILD in a MODIFY request. This is recommended whenever BrightStor CA-Allocate has been down for a long time (and significant DADSM activity has occurred), when volumes whose disk allocations are tracked by Quota are “lost” due to a hardware error and subsequently restored from backup, or whenever the Quota Group structure has been reorganized, thereby rendering the currently accumulated allocation statistics meaningless.

The QSYNC Process

The allocation statistics are maintained in two places: in real memory in the CSA Quota Table (CQT) and on disk in the Disk Quota Table (DQT). The QSYNC process ensures that the CQT and DQT contain the same information about the allocation statistics by periodically “re synchronizing” them. This is important when running BrightStor CA-Allocate in multi-CPU environments with shared disk volumes.

During start-up, the CQT is created by loading a copy of the DQT into real memory. As processing on the system continues, the QSYNCs periodically update the DQT with changes from the CQT. These changes were made to the CQT in real-time, following the completion of each successful DADSM operation. In a multi-CPU environment with shared disk volumes, a QSYNC also updates a CQT with changes that have been made in the DQT. The DQT changes came from the last QSYNC with the other CQTs in that multi-CPU environment. Figure 7-3 shows a simplified view of this process.

FIGURE 7-3 QSYNC Process



A QSYNC occurs at whatever frequency interval (SYNCINTERVAL) was specified in the QCONFIG parameter file. Additional QSYNCs are performed after a QREBUILD, during a REFRESH, and before uninstalling BrightStor CA-Allocate. You can also request a QSYNC at any time by specifying QSYNC in a MODIFY request. A SYSTEMS enqueue on QNAME STERLING and RNAME DQT-data-set-name is placed on the DQT for the duration of the QSYNC process.

In a single CPU environment, QSYNCs are important when accurate reports on the allocation statistics are needed. Both reporting vehicles—the QSTAT CLIST and the VDSQRPTS Batch job—get their information from the DQT, which depends on the QSYNCs for its information. QSYNC frequency intervals are of greater importance in multi-CPU environments because that is how changes in the allocation statistics on one system are communicated to the CQTs on the other systems.

Periodic synchronizations of the changes in the allocation statistics across all systems in a multi-CPU environment are critical for ensuring the accuracy of the information retrieved by the QUOTACHK Subroutine. QUOTACHK is used to determine whether a proposed DADSM Operation (for example, allocating a new data set) results in a group exceeding its predefined limit. QUOTACHK gets the current group's allocation statistics from the CQT, which depends on the QSYNCs to reflect changes in the allocation statistics on other systems.

How often QSYNCs should run depends on how the statistics are used. The QSYNC interval determines the accuracy of the quota allocation statistics. To deny allocations to groups of users who have exceeded their predefined limits requires far more accuracy than does daily reporting.

Water Mark Thresholds

Water marks are allocation thresholds that provide the user with a historical perspective of a quota group. Their purpose is to help determine whether or not a pre-defined quota group limit can be exceeded. Depending on which variables are used, the user can establish:

- QHIWATERMARK
- QLOWATERMARK
- Or a combination of both

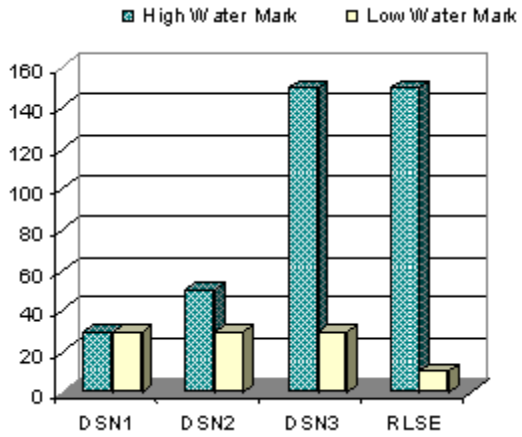
Water mark thresholds are valid until the next QREBUILD operation. This is because the QSYNC process of QREBUILD re-synchronizes the CQT, which in effect resets all water mark thresholds. They can also be re-initialized on demand by using the QRESETWM command, described in the section **QRESETWM** Subcommand in the chapter "**Operation**."

Note: To avoid having a QREBUILD reset the water marks, set PLSOPT8 to 'Y'. See **PLSOPT8** in the chapter "**Implementation**."

Operation

A quota group's high and low water marks are initialized to the allocation amount of the first data set associated with the quota group at the time the quota group is created. Thereafter, whenever a quota group's allocation amount changes, the new value is compared against both the high and low water marks. These high and low water marks fluctuate during the life cycle of a quota group, as shown in the bar chart below:

FIGURE 7-4 Quota High and Low Water Mark Chart



The high and low “water mark percentages” are calculated as a percent of the group's pre-defined quota limit. This is consistent with the QUOTAPERCENTS variable, representing the proposed total allocated amount (QUOTAALLOCS) as a percentage of a quota group's predefined limit (QUOTALIMITS), and with the “percents” included in the various BrightStor CA-Allocate reports.

Environments

Quota groups can be created and updated either in the QSCAN Environment during a Quota Table Rebuild operation or after a successful DADSM operation that was processed in the Quota Environment.

Quota Files

Quota utilizes five files:

1. Disk Quota Table
2. Quota Configuration File
3. QREBUILD Allocation Selection Routine
4. QSCAN Allocation Selection Routine
5. VDSPROG Allocation Selection Routine

Disk Quota Table

The Disk Quota Table (DQT) is a snapshot of the CSA Quota Table (CQT) taken at user-defined intervals. The QSYNC process is the camera. It is identified to the BrightStor CA-Allocate Started Task by either the PLSDQTDS parameter in the VKGPARMS member of PARMLIB or by the DISKQTBL DD statement in the Started Task itself. When BrightStor CA-Allocate is brought down, all changes in the CQT since the last QSYNC are propagated to the DQT. When BrightStor CA-Allocate is brought up, the CQT is created by the Started Task, which loads the DQT into CSA memory.

The DQT must be allocated with the attributes listed in Table 7-3. Space allocation amounts are discretionary. For example, on a 3390 device with a block size of 27,984 bytes, 5 tracks will support 2,640 Quota Groups. This file should be protected by security software so that only the Started Task can write on it. All other jobs should be able to read it.

TABLE 7-3 Required Attributes of the DQT

ATTRIBUTE	VALUE
Organization	Physical Sequential
Record Format	Fixed Blocked
Record Length	106 bytes
Block Size	Multiple of 106 bytes. As large as possible.
Primary Amount	user discretion
Secondary Amount	user discretion

The JCL to allocate the DQT is located in member ALLOCDQT in the Install Library.

Quota Configuration File

The Quota Configuration File contains the general parameters that control the operation of Quota. Its default member name is QCONFIG. **It is not an Allocation Selection Routine.** It has three types of statements:

- OPTIONS statement
- QUOTADEF statement
- QRESET statement

OPTIONS Statement

The OPTIONS statement specifies installation options. It is required and should be the first statement in the Quota Configuration File. Table 7-4 and Table 7-5 describe the associated keyword parameters.

TABLE 7-4 Quota's OPTIONS Statement

STATEMENT	KEYWORD PARAMETERS
OPTIONS	SYNCINTERVAL(NumberOfSeconds)
	[ASRQUOTAGROUPS({YES | NO})]
	[DEFAULTQUOTAGROUP({DEFAULT
	QUOTA GROUP | QuotaGroupName)]
	[QREBUILD({QREBUILD | MemberName})]
	[QSCAN({QSCAN | MemberName})]
	[SPACEUNIT({3380 | Unit Type})]

Note: See **TABLE 1-1 Notation Conventions used** within Tables, in the chapter "**Introduction**" of this guide, for a description of the notation used in the above table.

TABLE 7-5 OPTIONS Statement Keyword Parameters

STATEMENT	KEYWORD PARAMETERS
SYNCINTERVAL	The number of seconds the Quota Table Synchronization task (QSYNC) should sleep between Disk and CSA Quota Table synchronizations.
ASRQUOTAGROUPS	Option to permit creation of new Quota Groups by the VDSPROG and QSCAN ASRs. If ASRQUOTAGROUPS(NO) is specified, BrightStor CA-Allocate uses the default quota group whenever these ASRs select one that is not already defined in the Quota Table.
DEFAULTQUOTAGROUP	Default Quota Group if the VDSPROG ASR selects one not already defined in the Quota Table. Defaults to 'DEFAULT QUOTA GROUP'.
QREBUILD	The default value for the QREBUILD ASR is specified in the PLSQRBMB parameter in the VKGPARMS member of PARMLIB. The QREBUILD OPTION of the Quota Configuration File overrides the default name. As delivered, the ASR name is QREBUILD.
QSCAN	The default value for the QSCAN ASR is specified in the PLSQSCMB parameter in the VKGPARMS member of PARMLIB. The QSCAN OPTION of the Quota Configuration File overrides the default name. As delivered, the ASR name is QSCAN.
SPACEUNIT	The unit type to be used when quota limits are specified in TRK or CYL units in the QUOTADEF statements. The default is '3380'. Tracks and cylinders equal the following numbers of bytes for quota processing for these SPACEUNIT values:

SPACEUNIT	TRACK	CYLINDER
3350	19,069	572,070
3375	35,616	427,392
3380	47,476	712,140
3390	56,664	849,960
9340	48,280	724,200

All but one of the keyword parameters described in Table 7-5 have default values. The SYNCINTERVAL is the only required keyword parameter in the OPTIONS statement. SYNCINTERVAL does not specify how often QSYNC starts but rather the length of the "sleep" between the time the current QSYNC ends and the next one starts. To find out about the significance of this parameter see the section The QSYNC Process in the chapter "Quota."

QUOTADEF Statement

The QUOTADEF statement defines a Quota Group by specifying its name, quota limit, warning level, and parent. Table 7-6 lists the keyword parameters associated with the QUOTADEF statement.

TABLE 7-6 QUOTADEF Statement Keyword Parameters

STATEMENT	KEYWORD PARAMETERS
QUOTADEF	NAME(QuotaGroupName)
	[LIMIT(number{GB | KB | MB | PB | TB | TRK | CYL})]
	[WARNING(number[%])]
	[PARENT(QuotaGroupName)]

Note: See **TABLE 1-1 Notation Conventions used** within Tables, in the chapter "**Introduction**" of this guide, for a description of the notation used in the above table.

Only the NAME parameter is required. The other three are optional. You may need to modify one or more of these parameters for an existing Quota Group. To do so, do a QREBUILD after making any changes to the appropriate QUOTADEF statement.

QRESET Statement

You may need to reorganize the quota group structure and remove groups that are no longer required. The QRESET statement tells the Started Task to delete the contents of the current CQT and rebuild it. The QRESET statement is optional and, as shown in Table 7-7, has no keyword parameters.

TABLE 7-7 Quota's QRESET Statement Keyword Parameter

STATEMENT	KEYWORD PARAMETERS
QRESET	none

Manual versus Automatic Quota Group Definition

The Quota Groups used can be defined manually, automatically, or by a combination of the two. Any Quota Group, whether manually or automatically defined, can be modified by the QUOTADEF statement in the Quota Configuration File. Typically, manual quota group definition is used when there are few quota groups, and automatic quota group definition is used when it is possible to directly relate a portion of a data set name to a specific quota group. The generation of automatic quota group definitions is controlled by the ASRQUOTAGROUPS parameter of the OPTIONS statement in the Quota Configuration File. Two advantages to automatic quota group definition are:

- When there are many Quota Groups, it may not be feasible to maintain them manually.
- Quota can be installed quickly without expending much effort in defining the Quota Groups.

Sample Quota Configuration File

Figure 7-5 is a Quota Configuration File that can be used to gather the initial allocation statistics the first time Quota is brought up. The format is free-form and blank lines are permitted. However, if a statement spans more than one line, the break cannot occur in the middle of a keyword or value pair and must be preceded by at least one space followed by a plus sign (+).

FIGURE 7-5 Sample Quota Configuration File

```
File   Edit   Confirm   Menu   Utilities   Compilers   Test   Help
------------------------------------------------------------------------
EDIT               .PARMLIB(QCONFIG) - 01.19               Columns 00001 00072
Command ==> ____________________________________________________ Scroll ==> CSR

****** *************************** Top of  Data ********************************
000001  OPTIONS    SYNCINTERVAL(60) ASRQUOTAGROUPS(YES)                  +
000002             DEFAULTQUOTAGROUP(MISCELLANEOUS GROUP)                +
000003             QREBUILD(QREBUILD) QSCAN(QSCAN) SPACEUNIT(3380)
000004  QUOTADEF   NAME(MAIN STREET COMPUTER CENTER) LIMIT(2000CYL) WARNING(70%)
000005  QUOTADEF   NAME(AAA DEPT) LIMIT(700MB) WARNING(71%)              +
000006             PARENT(MAIN STREET COMPUTER CENTER)
000007  QUOTADEF   NAME(BBB DEPT) LIMIT(600MB) WARNING(65%)              +
000008             PARENT(MAIN STREET COMPUTER CENTER)
000009  QUOTADEF   NAME(CCC DEPT) LIMIT(500MB) WARNING(80%)              +
000010             PARENT(MAIN STREET COMPUTER CENTER)
000011  QUOTADEF   NAME(AAAMAH) PARENT(AAA DEPT) LIMIT(200MB) WARNING(80%)
000012  QUOTADEF   NAME(AAAMAI) PARENT(AAA DEPT) LIMIT(210MB) WARNING(81%)
000013  QUOTADEF   NAME(AAAMAJ) PARENT(AAA DEPT) LIMIT(220MB) WARNING(82%)
000014  QUOTADEF   NAME(BBBMAK) PARENT(BBB DEPT) LIMIT(230MB) WARNING(83%)
000015  QUOTADEF   NAME(BBBMAL) PARENT(BBB DEPT) LIMIT(240MB) WARNING(84%)
000016  QUOTADEF   NAME(BBBMAM) PARENT(BBB DEPT) LIMIT(250MB) WARNING(85%)
000017  QUOTADEF   NAME(CCCMAN) PARENT(CCC DEPT) LIMIT(260MB) WARNING(86%)
000018  QUOTADEF   NAME(CCCMAO) PARENT(CCC DEPT) LIMIT(270MB) WARNING(87%)
000019  QUOTADEF   NAME(CCCMAP) PARENT(CCC DEPT) LIMIT(280MB) WARNING(88%)
```

After the initial allocation statistics have been collected, the QUOTADEF statements should be removed from the active Quota Configuration File. During a REFRESH operation, the active configuration file is recompiled. If either a QUOTADEF or QRESET statement is found, BrightStor CA-Allocate assumes that a QREBUILD operation is required and automatically starts one.

Allocation Selection Routines and Environments

The section “Implementation” describes available Allocation Selection Routine (ASR) statements, constants, functions, and variables. The ASRs for Quota are essentially the same as those for the Allocation Manager. The two selectable units share the VDSPROG ASR. Two ASRs are unique to Quota's QREBUILD process: QREBUILD and QSCAN. Three of the fourteen Environments apply. The QREBUILD and QSCAN Environments are used by the QREBUILD process. The QUOTA Environment is used when tracking proposed and actual changes to the allocation statistics.

QUOTA ASR and SMS Constructs

Sample ASR code:

```
IF &VAMENVIR NE 'QUOTA' THEN EXIT CODE(0)
WRITE 'DATA CLASS = &DATACLAS'
WRITE 'MANAGEMENT CLASS = &MGMTCLAS'
WRITE 'STORAGE CLASS = &STORCLAS'
WRITE 'STORAGE GROUP = &STORGRP'
IF &STORCLAS = 'DASD80' THEN SET &QUOTAGRP = 'STORCLAS80'
CALL QUOTACHK
WRITE 'QUOTAGROUP = &QUOTAGROUP'
EXIT CODE(0)
```

QREBUILD ASR and Environment

The QREBUILD ASR and Environment filter volumes for Quota processing when the Quota Table is being built or rebuilt from data obtained by scanning the VTOCs during the QREBUILD process. For each volume that is online during the Quota Table rebuild, the QREBUILD ASR and Environment receive control one time. An example of a QREBUILD ASR is:

```
FILTLIST &QVOLS INCLUDE(SYS*,JES*)       /* EXCLUDE THESE VOLUMES */
IF &VOLSER EQ &QVOLS THEN EXIT CODE(4)   /* FROM QUOTA PROCESSING */
EXIT CODE(0)                             /* ELSE INCLUDE THIS VOL IN VTOC SCAN */
```

Sample QREBUILD ASR:

```
IF &STORGRP = '' THEN EXIT CODE(4)       /* CONSIDER ONLY SMS VOLUMES */
WRITE 'SMS STORGRP = &STORGRP'
WRITE 'SMS VOLSER = &VOLSER'
EXIT CODE(0)
```

For QREBUILD, &STORGRP contains the SMS storage group name that the volume belongs to (non-SMS VDSTORPG storage groups are not returned to the &STORGRP variable). For QUOTA and QSCAN, &STORGRP contains the SMS storage group name of the volume that the data set is being allocated to, or currently resides on.

QSCAN ASR and Environment

The QSCAN ASR and Environment determine which Quota Group is accountable for which data set's allocation amount when the initial statistics are being captured. They receive control one time for each data set found on each volume selected for processing during the QREBUILD process. An example of a QSCAN ASR is:

```
SET &QUOTAGROUP = &HLQ                   /* QUOTA GROUP IS HIGH-LEVEL QUAL   */
SET &C0 = &SUBSTR(1:3,&HLQ)              /* IF NEW, PARENT IS 1ST THREE      */
SET &QPARENTIFNEW = &C0                  /* CHARACTERS OF HIGH-LEVEL QUAL    */
SET &QLIMITIFNEW = 100 MB                /* IF NEW, LIMIT IS 100 MB          */
SET &QWARNINGIFNEW = 80                  /* IF NEW, WARN AT 80% OF LIMIT     */
```

VDSPROG ASR and the QUOTA Environment

The QUOTA Environment is entered for every DADSM data set operation for all types of data sets, both VSAM and non-VSAM, including SMS-managed. The ASR logic in the QUOTA Environment:

- Determines whether the volume on which the DADSM operation is to take place is subject to Quota processing. If not, the ASR exits.
- Determines which group is to be accountable for the amount involved in the DADSM operation (CREATE, EXTEND, RELEASE, RENAME, or SCRATCH).
- Calls the QUOTACHK internal routine, which has two functions. The first function is determining what impact the proposed DADSM operation has on the applicable group's allocation statistics. The second is building the record to update the CQT copy of the allocation statistics if the operation is successful.
- Decides whether to allow the operation, based on information returned by QUOTACHK, and optionally write messages to the user.
- Always use the &DSN variable for tracking DFSMShsm activities.

The QUOTA Environment ASR code is kept in the VDSPROG ASR, which also contains the code for the other eleven Environments applicable to DADSM operations. These other environments comprise the BrightStor CA-Allocate Allocation Manager feature and are described in detail in the section **Environments** in the chapter "**Concepts and Facilities**." Figure 7-6 shows an example of a QUOTA Environment ASR.

FIGURE 7-6 Sample QUOTA ASR (Page 1 of 3)

```
Menu     Utilities    Compilers    Help
------------------------------------------------------------------------
BROWSE          .ASR.PARMLIB(QUOTA1) - 01.02           Line 00000000 Col 001 080
Command ==> ____________________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************
IF &VAMENVIR = QUOTA THEN
    DO
       IF (&VOLSER EQ SYS*) OR (&VOLSER EQ JES*) THEN EXIT CODE(0)
       COPYBOOK 'QSCAN'
       CALL QUOTACHK
       IF (&QUOTAFUNC = 'RELEASE') OR (&QUOTAFUNC = 'SCRATCH')
          THEN EXIT CODE(0)
       IF &QUOTAFUNC = 'RENAME' THEN SET &NEWQUOTAGROUP = &NEWNAMEHLQ
       If &QUOTACC = 0                      /*  UNDER WARNING LIMIT ?  */
            THEN EXIT CODE(0)               /*  THEN EXIT ASR NOW      */
/*                                                                      */
/*  GROUPS WITHIN THEIR WARNING LIMIT OR THAT HAVE ALREADY EXCEEDED     */
/*                                                                      */
       COPYBOOK 'QUOTA2'
    END
******************************** Bottom of  Data ********************************
```

```
Menu     Utilities    Compilers    Help
------------------------------------------------------------------------
BROWSE          .ASR.PARMLIB(QUOTA2) - 01.01           Line 00000000 Col 001 080
Command ==> _____________________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************
       SET  &C0 = &QUOTANAMES(1)             /*    CHILD QUOTA GROUP     */
       SET  &N0 = &QUOTALIMITS(1)            /*    ITS QUOTA LIMIT       */
       SET  &N1 = &QUOTAALLOCS(1)            /*  TOTAL ALLOCATED SPACE   */
       SET  &N2 = &QUOTAPERCENTS(1)          /*    PERCENT OF LIMIT      */
       COPYBOOK 'QUOTA3'                     /*  ISSUE WARNING MESSAGE*/
       SET  &C0 = &QUOTANAMES(2)             /*    CHILD QUOTA GROUP     */
       SET  &N0 = &QUOTALIMITS(2)            /*    ITS QUOTA LIMIT       */
       SET  &N1 = &QUOTAALLOCS(2)            /*  TOTAL ALLOCATED SPACE   */
       SET  &N2 = &QUOTAPERCENTS(2)          /*    PERCENT OF LIMIT      */
       COPYBOOK 'QUOTA3'                     /*  ISSUE WARNING MESSAGE*/
******************************** Bottom of  Data ********************************
```

```
Menu     Utilities    Compilers    Help
------------------------------------------------------------------------
BROWSE          .ASR.PARMLIB(QUOTA3) - 01.01           Line 00000000 Col 001 080
Command ==> _____________________________________________________ Scroll ==> CSR

******************************** Top of  Data ********************************
    WRITE 'QUOTA GROUP... &C0 QUOTA LIMIT... &N0 KB'
    WRITE 'ALLOCATION AMOUNT... &N1 KB (AT &N2% OF LIMIT)'
    IF &QUOTASTATS(1) = 4                  /* WITHIN WARNING LIMITS ?  */
       THEN DO
              WRITE 'YOUR GROUP IS NEAR ITS QUOTA LIMIT'
            END
       ELSE DO                             /* MUST HAVE EXCEED LIMIT   */
              WRITE 'YOUR GROUP HAS EXCEEDED ITS QUOTA LIMIT.'
              WRITE 'PLEASE REDUCE YOUR DISK SPACE SO THAT YOU ARE'
              WRITE 'WITHIN YOUR QUOTA LIMITS. THANK YOU.'
            END
******************************** Bottom of  Data ********************************
```

Relationship Between the QUOTA, QREBUILD, and QSCAN ASRs

When tracking changes in the allocation statistics, it is important to use the same criteria used to initialize or re-initialize the Quota Table. The accuracy of these statistics depends on it. Tracking changes to the allocation statistics is meaningless unless you know at the start how much space is allocated and by which group. Note the use of the COPYBOOK statement in Figure 7-6. It references the actual QSCAN ASR used by the QREBUILD process. This is the most efficient way to ensure Quota consistency.

Reporting Options

Quota has the following reporting options:

- QSTAT CLIST
- VDSQRPTS Batch Job
- VDSQRPT2 Batch Job
- VDSQRPT3 Batch Job
- Online
- Batch Jobs

Both reporting vehicles get their information from the DQT. The accuracy of the information retrieved by both reporting options depends on how much DADSM Activity has occurred since the last QSYNC. Ad-hoc reporting is possible with the availability of the DQT record description that is distributed in the AVDSMAC DDDEF data set.

QSTAT CLIST

QSTAT is a TSO CLIST that shows the status of a specific Quota Group. It is supplied on the installation tape as member QSTAT in the INSTALL PDS. It must be copied to the standard SYSPROC library so that it can be used by all TSO users. The following is the QSTAT CLIST as it is distributed:

```
PROC 0 GROUP(UNSPECIFIED)
ALLOC FI(DISKQTBL) DA('VAM.Rvrm.DISKQTBL') SHR REUSE
CALL 'VAM.Rvrm.LOADLIB(VAMQST1)' '&GROUP'
FREE FI(DISKQTBL)
END
```

Tailor the CLIST to reference your Disk Quota Table and Load Library. The program VAMQST1 displays the current status of the quota group requested and the status of all its parents. If the VAMQST2 program is used, only the current status of the quota group is displayed.

The syntax for using QSTAT is:

```
QSTAT GROUP('Quota Group Name')
```

The Quota Group Name must be enclosed in single quotation marks if it contains embedded blanks. Figure 7-7 shows sample output from QSTAT.

FIGURE 7-7 Sample Output from QSTAT TSO Command

```
CA-Allocate Quota Status Information as of 06/13/2001 08:29:23

QUOTA GROUP = TSOST LPAR
    QUOTA LIMIT.............29.00 GB                    31,138,512,896 BYTES
    SPACE ALLOCATED.........1.45 MB (0%)                     1,519,232 BYTES
    SPACE REMAINING.........29.00 GB (100%)            31,138,512,896 BYTES
    HIGH WATER MARK.........1.45 MB (0%)              at 06/13/2001 08:28:14
    LOW WATER MARK..........1.45 MB (0%)              at 06/13/2001 08:28:14
    PARENT QUOTA GROUP.....COMPUTER ASSOCIATES

    QUOTA GROUP = COMPUTER ASSOCIATES
    QUOTA LIMIT.............100.00 GB                  107,374,182,400 BYTES
    SPACE ALLOCATED.........1.45 MB (0%)                     1,519,232 BYTES
    SPACE REMAINING.........100.00 GB (100%)           107,374,182,400 BYTES
    HIGH WATER MARK.........1.45 MB (0%)              at 06/13/2001 08:28:14
    LOW WATER MARK..........1.45 MB (0%)              at 06/13/2001 08:28:14
    PARENT QUOTA GROUP.....NONE

---End of CA-Allocate Quota Status Information---
***
```

Batch Reports

The batch report jobs use stand-alone programs to produce detailed reports on all Quota Groups. They are supplied in the INSTALL library as members:

VDSQRPTS - quota limit, space allocated, and space remaining

VDSQRPT2 - space allocated, high water mark, and low water mark

VDSQRPT3 - limit, space allocated, space remaining, and high/low water marks

Figure 7-8, Figure 7-9, and Figure 7-10 show samples of each report.

FIGURE 7-8 Sample VDSQRPTS Output, Quota Groups Sorted by Name

```
Menu     Utilities    Compilers    Help
------------------------------------------------------------------------------------------------------------
BROWSE                                                                  Line 00001374 Col 001 132
Command ==> ____________________________________________________________________________ Scroll ==> CSR
------------------------------------------------------------------------------------------------------------
STATUS as of 02/03/2001 22:39:35          CA-Allocate QUOTA GROUP STATUS REPORT #1            PAGE   11
                                             SORTED BY DESCENDING SPACE ALLOCAED
                                                     ALLOCATED 5.3.0

      QUOTA GROUP NAME/PARENT          QUOTA LIMIT   SPACE ALLOCATED     %    SPACE REMAINING      %
-----------------------------------   ------------   ---------------   ----   ---------------    ----
LAB** DSETS ON VOL SSL802                500.00 KB         417.27 KB     83%          82.73 KB      17%
SSL802
-----------------------------------   ------------   ---------------   ----   ---------------    ----
LAB** DSETS ON VOL SSL801                500.00 KB         417.27 KB     83%          82.73 KB      17%
SSL801
-----------------------------------   ------------   ---------------   ----   ---------------    ----
CSL** DSETS ON VOL SSL805                500.00 KB         324.54 KB     65%         175.46 KB      35%
SSL805
-----------------------------------   ------------   ---------------   ----   ---------------    ----
ALL OTHER* DSETS ON VOL SY9SY1           500.00 KB         278.18 KB     56%         221.82 KB      44%
SY9SY1
-----------------------------------   ------------   ---------------   ----   ---------------    ----
STA* DSETS ON VOL SY9SY1                  50.00 KB         278.18 KB      1%          49.73 KB      99%
SY9SY1
-----------------------------------   ------------   ---------------   ----   ---------------    ----
ALL OTHER* DSETS ON VOL SMS903           500.00 KB         276.60 KB     55%         223.32 KB      45%
```

FIGURE 7-9 Sample VDSQRPT2 Output, Quota Groups Sorted by Allocation Amount

```
Menu     Utilities    Compilers    Help
------------------------------------------------------------------------------------------------------------
BROWSE                                                                        Line 00000720 Col 001 132
Command ==> ____________________________________________________________________________ Scroll ==> CSR
------------------------------------------------------------------------------------------------------------
STATUS as of 02/03/2001 22:39:35          CA-Allocate QUOTA GROUP STATUS REPORT #2            PAGE   13
                                             SORTED BY QUOTA GROUP NAME
                                                   ALLOCATED 5.3.0

      QUOTA GROUP NAME/PARENT          SPACE ALLOCATED        HIGH WATER MARK          LOW WATER MARK
---------------------------------    ----------------    --------------------    --------------------
VAM* DSETS ON VOL SMS901               221.83MB 222%          221.83MB   222%          221.83MB   222%
SMS901                                                      at 01/29/2001 10:28     at 01/29/1996 10:28
---------------------------------    ----------------    --------------------    --------------------
VAM* DSETS ON VOL SSL801               151.90MB 152%          151.90MB   152%          151.90MB   152%
SSL801                                                      at 01/29/2001 10:28     at 01/29/1996 10:28
---------------------------------    ----------------    --------------------    --------------------
VAM* DSETS ON VOL SSL802               174.13MB 174%          174.13MB   174%          174.13MB   174%
SSL802                                                      at 01/29/2001 10:28     at 01/29/1996 10:28
---------------------------------    ----------------    --------------------    --------------------
VAM* DSETS ON VOL SSL803               362.03MB 362%          362.03MB   362%          362.03MB   362%
SSL803                                                      at 01/29/2001 10:28     at 01/29/1996 10:28
---------------------------------    ----------------    --------------------    --------------------
VAM* DSETS ON VOL SSL804                20.28MB  20%           20.28MB    20%           20.28MB    20%
SSL804                                                      at 01/29/2001 10:28     at 01/29/1996 10:28
---------------------------------    ----------------    --------------------    --------------------
VAM* DSETS ON VOL SSL805               149.10MB 149%          149.10MB   149%          149.10MB   149%
```

FIGURE 7-10 Sample VDSQRPT3 Output, Quota Groups Sorted by Percentage of Limit

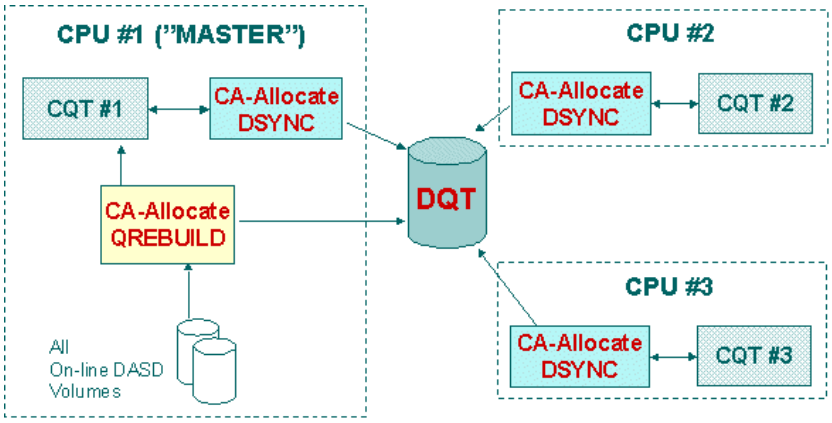
```
Menu     Utilities    Compilers    Help
------------------------------------------------------------------------------------------------------------
BROWSE                                                                        Line 00000240 Col 001 132
Command ==> ____________________________________________________________________________ Scroll ==> CSR
------------------------------------------------------------------------------------------------------------
STATUS as of 02/03/2001 22:39:35        CA-Allocate QUOTA GROUP STATUS REPORT #3            PAGE    5
                                             SORTED BY QUOTA GROUP NAME
                                                 ABC WIDGET COMPANY

      QUOTA GROUP NAME/PARENT      LIMIT       SPACE ALLOCATED    SPACE REMAINING            HIGH/LOW WATER MARKS
---------------------------------  ----------  ----------------  ---------------  ----------------------------------
ISP** DSETS ON VOL SMSSTG          750.00MB          3.24MB   0%  746.76MB  100%     3.24MB     0% at 01/29/2001 10:20
SMSSTG                                                                              3.24MB     0% at 01/29/2001 10:20
---------------------------------  ----------  ----------------  ---------------  ----------------------------------
ISP** DSETS ON VOL SMS801          750.00MB        108.94MB  15%  641.06MB   85%   108.94MB    15% at 01/29/2001 10:20
SMS801                                                                            108.94MB    15% at 01/29/2001 10:20
---------------------------------  ----------  ----------------  ---------------  ----------------------------------
ISP** DSETS ON VOL SMS802          750.00MB        314.40MB  42%  435.60MB   58%   314.40MB    42% at 01/29/2001 10:20
SMS802                                                                            314.40MB    42% at 01/29/2001 10:20
---------------------------------  ----------  ----------------  ---------------  ----------------------------------
ISP** DSETS ON VOL SMS803          750.00MB         96.48MB  13%  653.52MB   87%    96.48MB    13% at 01/29/2001 10:20
SMS803                                                                             96.48MB    13% at 01/29/2001 10:20
---------------------------------  ----------  ----------------  ---------------  ----------------------------------
ISP** DSETS ON VOL SMS901          750.00MB        301.97MB  40%  448.03MB   60%   301.97MB    40% at 01/29/2001 10:20
SMS901                                                                            301.97MB    40% at 01/29/2001 10:20
---------------------------------  ----------  ----------------  ---------------  ----------------------------------
ISP** DSETS ON VOL SMS902          750.00MB        264.14MB  35%  485.86MB   65%   264.14MB    35% at 01/29/2001 10:20
```

Technical Details

Running Quota Across Multiple CPUs Sharing Disk Volumes

In a multi-CPU environment where disk volumes are shared, Quota can effectively track the allocation statistics on the shared disk volumes. It can also concurrently track those volumes that are not shared. The only limitation is that the single DQT needs to reside on one of the shared volumes. This enables each QSYNC to update the DQT with the changes that have occurred on its system and, in turn, to receive information from the DQT about the changes that have occurred on the other systems. Figure 7-11 shows a simplified view of this process.

FIGURE 7-11 Proper Way of Running Quota in Multi-CPU Environments



The communication between the three CPUs in Figure 7-11 occurs as follows:

1. A Started Task runs on each CPU.
2. The necessary operating system hooks are installed on all CPUs.
3. A copy of the Disk Quota Table (DQT), named the CSA Quota Table (CQT), is maintained on each CPU.
4. The QSYNC operation is done on all CPUs. The frequency of this operation for each CPU is determined by the QSYNCINTERVAL parameter on the OPTIONS statement in the QCONFIG file on each CPU.
5. The QREBUILD operation is done on only one CPU, and the result is copied to DQT replacing its contents. This CPUs Started Task is designated as the "master." The Started Tasks running on the other CPUs automatically load CQT changes from the DQT after the "master" Started Task writes them to the DQT.

Note: In a shared DASD environment, a QREBUILD must not be done on any CPU other than the "Master", as shown in Figure 7-11. If QREBUILD is allowed to execute on a secondary CPU, and the secondary CPU does not have access to all DASD, the DQT may only report on a subset of your DASD volumes.

QUOTA and PLSQFACT

When PLSQFACT (K) is specified, all numeric values in quota are expressed in units of kilobytes. If a DADSM operation resulted in an allocation of 3120 bytes, this would be converted to kilobytes and rounded up. In this case, the resulting allocation value (as tracked by QUOTA) would be 4 kilobytes.

Running with PLSKBYTE (1000) instead of the default (1024) will introduce a loss of precision resulting in the quota allocation amounts overstating what will actually be allocated. If running in fail mode, this may result in allocations being erroneously denied because the quota limit has been, or would otherwise be, exceeded.

If a PARMREF is done, which changes the value of PLSQFACT, a subsequent QREBUILD is necessary to ensure the integrity of the QUOTA statistics.

Logically Disabling Quota Processing

Occasionally it can be necessary to disable all Quota processing activities temporarily. One example is during a Disaster Recovery operation, when the goal is to get the operating system and all critical applications up and running as quickly as possible. In a crisis situation, it may not be feasible to adhere to normal standards.

To run BrightStor CA-Allocate with Quota effectively disabled, the following is needed:

1. The DISKQTBL DD statement in the Started Task must point to a valid DQT allocated as detailed in Table 7-3, "Required Attributes of the DQT."
2. The QSCAN ASR need only contain an "EXIT CODE(0)" statement.
3. The QREBUILD ASR need only contain an "EXIT CODE(4)" statement.
4. The QCONFIG configuration file need only contain the sample QCONFIG from the install library created during the SMP/E install process, with one modification: SYNCINTERVAL(99999).
5. The ASRs and Configuration Files listed in (2), (3), and (4) must be in the partitioned data set referenced by the PLSPRGDS variable within VKGPARM in the PARMLIB.

With the above, Quota is logically disabled. No tracking is done. The QSYNC Task runs one time every 28 hours.

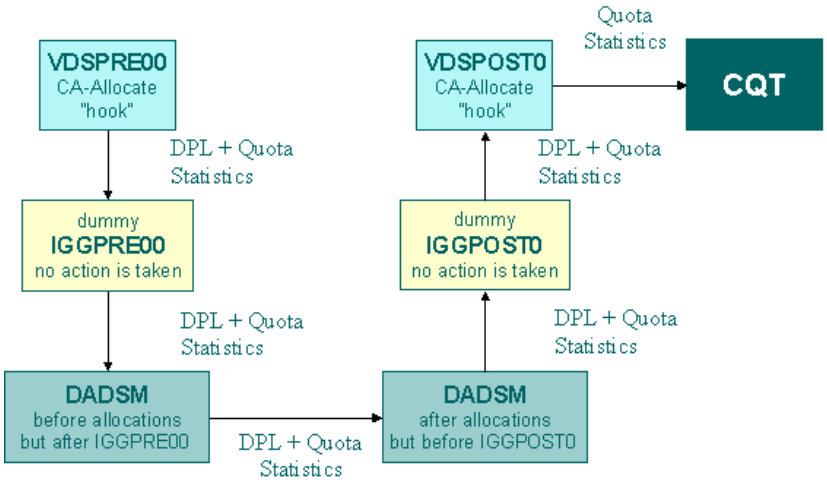
Exposures of Sharing IGGPRE00 and IGGPOST0

Quota is not unique in its use of the standard DADSM exits IGGPRE00 and IGGPOST0. Other program products use either one or both of them. Special care needs to be taken in bringing up BrightStor CA-Allocate on systems where it co-exists with one of these other program products. An installation's MVS Systems Programming Staff knows of any other products.

The QUOTACHK Subroutine, called from the QUOTA Environment, has two functions. The first function is to determine the impact a proposed change has on a group's allocation statistics. The second is to build the record that is used to update the CQT copy of the allocation statistics if the proposed DADSM operation is successful.

This record is built at “VDSPRE00-time.” The update occurs at “VDSPOST0-time.” DADSM has a parameter list that passes information from its IGGPRE00 exit to its IGGPOST0 exit. At VDSPRE00, after it has built the update record, Quota puts the location of that record in the IEXRSVWD field in the DADSM parameter list (DPL). At VDSPOST0, Quota retrieves the location of the update record from this field in the DPL, then updates the CQT with it. Figure 7-12 shows a simplified view of this process.

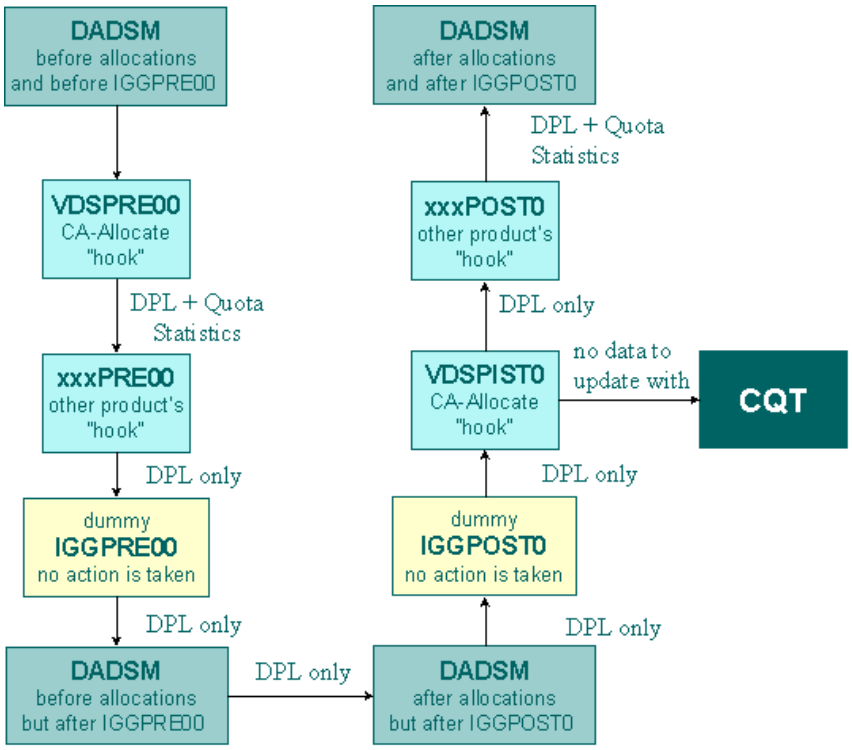
FIGURE 7-12 How Quota Updates the CQT



The exposure arises because other program products can also use the IEXRSVWD field in the DPL. Quota has a “good neighbor” policy. At “VDSPRE00-time,” it saves the original contents of this field in the update record. At “VDSPOST0-time,” it restores it. Other program products cannot be relied on to do so. If they do not, and they are “hooked” before BrightStor CA-Allocate, the Statistics are lost. Even if other program products practice the “good neighbor” policy of saving the original contents of this field and restoring it when they are done, there is still an exposure.

For example, consider the situation depicted in Figure 7-13. Suppose another program product, with hooks at both places, also uses this field in the DPL to pass information. If the installation order is IGGPRE00-xxxPRE00-VDSPRE00 and IGGPOST0-VDSPOST0-xxxPOST0, the Statistics are lost. When Quota gets the DPL at “VDSPOST0-time,” the IEXRSVWD field does not contain the pointer to the Statistics. It contains whatever information the other program product's hook put there at “xxxPRE00-time.” The way to avoid such “exposures” is to adjust the startup times of both products to ensure that one of them completes its installation before the other one begins.

FIGURE 7-13 How Quota’s Allocation Statistics Can Be “Lost”



RENAME and MANAGEMENT CLASS

Renaming a SMS data set (using ISPF rename or IDCAMS ALTER NEWNAME) re-drives the ACS MANAGEMENT CLASS routine. If the ACS routine changes the management class during QUOTAFUNC=RENAME time, the &MC variable does not reflect the new management class name. However, the IDCAM ALTER command also allows a new MGMTCLAS to be specified on a rename. In this case, during QUOTAFUNC=RENAME, the &MC variable has the new management class name.

API for Dynamically Changing Quota Limits

You can change the Quota Limit of an existing Quota Group without incurring the overhead associated with a quota table rebuild (QREBUILD) operation. QLIM, QPARENT, and QWPCT are initially specified when the QGRP is created. QGRPs (Quota Group) can be created three different ways:

- Explicitly defining them in the QCONFIG file
- Dynamically defining them in the QSCAN Environment when STATs are being tracked.
- Dynamically defining them in the QUOTA ASR Environment when STATs are being tracked.

The first two ways are only used in QREBUILD operations. The last is invoked for all DADSM operations. The only current way to change the QLIM (Quota Limit), QPARENT (Parent Quota Group), and QWPCT (Quota warning percent) attributes of an existing QGRP is by explicitly defining the QGRP, with the new attributes, in a QUOTADEF statement in the QCONFIG file prior to initiating a QREBUILD operation.

The VAMSRV10 API provides a second way for the QLIM and QWPCT of an existing QGRP to be changed and a fourth way for a new QGRP to be created. This API can be used to either change the QLIM or QWPCT of an existing QGRP or add a new QGRP. This API can be directly called from batch jobs, or indirectly invoked from either TSO CLISTs or REXX programs.

VAMSRV10 must reside in an APF-authorized library. This requires that any caller that directly invokes the API, such as a TSO CLIST or REXX program, be running APF-authorized. Callers who are unable to run APF-authorized are limited to invoking the API indirectly by generating and submitting the JCL that calls the API from an “EXEC PGM=VAMSRV10” statement in a batch job.

The API updates the CQT. These changes are propagated to the DQT with the next QSYNC, at the same time the STATs captured from the DADSM allocations intercepted in the QUOTA Environment are currently being propagated. From the DQT, these changes are propagated across all applicable LPARs with the next the QSYNC invocation on those systems.

Invoking the API

From a Batch job:

```
EXEC PGM=VAMSRV10,PARM='QGRP,QLIM,QWPCT,QPIFNU'
```

From an APF-authorized REXX program:

```
uparms = 'QGRP,QLIM,QWPCT,QPIFNU'
address linkmvs 'VAMSRV10 uparms'
if rc ne 0 then say 'ERROR return code = ' rc
```

From a nonAPF-authorized REXX program or TSO CLIST:

Caller's who are unable to run APF-authorized are limited to invoking the API indirectly by generating and submitting the JCL that calls the API from an "EXEC PGM=VAMSRV10,PARM=…" statement in a batch job. If the EXEC statements are kept in and retrieved from partitioned data set member, then that member can be used to regenerate the quota specifications if the DQT gets lost.

Calling Parameters

The similarity between the parameter names to existing variables is intentional. Except where noted, the documented definition, length, format, restrictions, limitations, and default values of the existing variables also apply to the analogous new calling parameter.

- Parameter #1 (required)

  `QGRP`

- Parameter #2 (optional)

  `QLIM (TRK and CYL not supported)`

- Parameter #3 (optional)

  `QWPCT`

- Parameter #4 (optional)

  `QPIFNU`

TABLE 7-8 Return Codes from VAMSRV10

RC	Description
0	CSA Quota Table successfully updated with specified changes
1	Unable to allocate storage for VAMSRV10's Dynamic Storage Area
2	VAMSRV10 not being run/invoked from an APF-Authorized library/program
3	Unable to restore caller's AMODE
4	BrightStor CA-Allocate is not installed
5	BrightStor CA-Allocate is installed, but without the Quota Selectable Unit
6	Unable to locate BrightStor CA-Allocate's DTICVT Control Block
7	Register 1 upon entry did not contain pointer to calling parameter list
8	Missing calling parameter list address
9	Invalid address for calling parameter list, VSMLOC RC= xx
10	Zero length for input parameter list
11	Missing QLIM input parameter
12	Non-numeric or zero value (xxx) specified for QLIM
13	Neither QLIM or QWPCT changes explicitly/implicitly requested for existing QGRP
14	Unit specified for QLIM (xxx) not 'BY' or 'KB' or 'MB' or 'GB' or 'TB'
15	QLIM amount specified exceeds current maximum value for limit

RC	Description
16	Non-numeric value (xxx) specified for QWPCT
17	Length of QWPCT specified > 3
18	QWPCT specified > 100 (%)
19	Length of QGRP specified > 50
20	Have 'lost' the address of the DTICVT control block
21	Missing address of the CSA Quota table (CQT)
22	CQT attribute check failed
23	Have 'lost' the CQTs address
24	VDSQSRCH's RC was zero, but it didn't retrieve the CQTREC address for the existing QGRP
25	Have 'lost' the length of the new QGRP
26	Unable to locate CQTREC for QPFINU parent-quota-group-name
27	Unable to locate QGRP ID number for QPFINU parent-quota-group-name
28	CQT0GRP# value in the CQT is zero
29	Non-zero return code (RC) from VDSQINSQ
30	VDSQINSQ's RC was zero, but it didn't insert the CQTREC for the new QGRP
31	Length of QPIFNU specified > 50
32	Have 'lost' the address of the DTICVT control block
33	BrightStor CA-Allocate has been taken out since this program has started running
34	DTICVT's missing the address of the Quota Common Control Block
35	Unrecoverable logic error in VAMSRV10

Appendix

A

Diagnostics

Activating a Trace

To trace the execution of the active Allocation Selection Routine (ASR) and to receive additional information throughout the volume selection routines, include the following JCL statement in a batch job step:

```
//VDSDIAGS DD DUMMY
```

When running under TSO, turn on this trace by entering the following TSO command:

```
ALLOC FI(VDSDIAGS) DUMMY
```

To deallocate a TSO trace, use the following command:

```
FREE FI(VDSDIAGS)
```

Every executable line of the current ASR is loaded into memory along with the compiled code. The compiler inserts a pseudo-instruction at the beginning of every source line. The instruction checks for the indication that the trace flag is on. If it is, the corresponding source line is sent to the user's terminal or job log.

In addition, the first time a variable is retrieved, BrightStor CA-Allocate checks the trace flag. When the trace flag is on, message VAM0100 displays the name of the obtained variable and its current value on the terminal or job log.

The trace facility monitors the progress of the ASR and the contents of each variable. For example, trace output can show the following:

```
IF &DSN = SYS1.** THEN
VAM0100 DSN = SYS1.PROCLIB
SET &STORGRP = 'SYSTEM'
VAM0100 STORGRP =
```

Note: The STORGRP= variable is empty when it is retrieved for the first time. After the SET statement executes, the STORGRP= variable contains 'SYSTEM'.

Note: VDSDIAGS provides limited diagnostics and will not show ACS activity, nor the complete set of diagnostics that Customer Support would require for working on a problem that you report. To obtain complete diagnostics, issue the command F VAM.DIAGS=jobname (batch jobname, STC name, or TSO USERID to be traced). Once the desired diagnostics have been captured, tracing can be deactivated by the command F VAM.DIAGS=. The VKGPARM, PLSDIAGS can also be used to specify the jobname to be traced. This would require a PARMREF after making the VKGPARMS update.

Appendix

B

Bypassing Allocations

If you need to bypass BrightStor CA-Allocate for all allocations in a step, allocate DDNAME VDSBYPAS to the step or TSO session, as shown below:

In a batch job step:

```
//VDSBYPAS DD DUMMY
```

For a TSO session:

```
ALLOC FI(VDSBYPAS) DUMMY
```

Note: DD statements cannot always be “seen” in the ACS environment. To bypass all Allocate processing for a job, including the ACS environment, it will be necessary to modify your ASR. For example, add the following at the beginning of your ASR:

```
IF &JOBNAME = 'MYJOB' THEN EXIT CODE(0)
```

Appendix

C

Dormant Operation

BrightStor CA-Allocate can operate in a dormant mode, meaning it is installed and fully functional, but affecting only selected jobsteps and TSO sessions. Dormant mode is useful in many situations: beta release testing, installation testing, testing new ASR logic, and so on.

To turn on BrightStor CA-Allocate for a jobstep, allocate the ddname specified in the PLSDRMNT parameter. As shipped PLSDRMNT is #VAMTST#.

After the operating mode is changed to DORMANT, the #VAMTST# DD name needs to be added to any jobstep or TSO session to be controlled by BrightStor CA-Allocate.

For a batch job step, use:

```
//#VAMTST# DD DUMMY
```

For a TSO session, use:

```
ALLOC FI(#VAMTST#) DUMMY
```

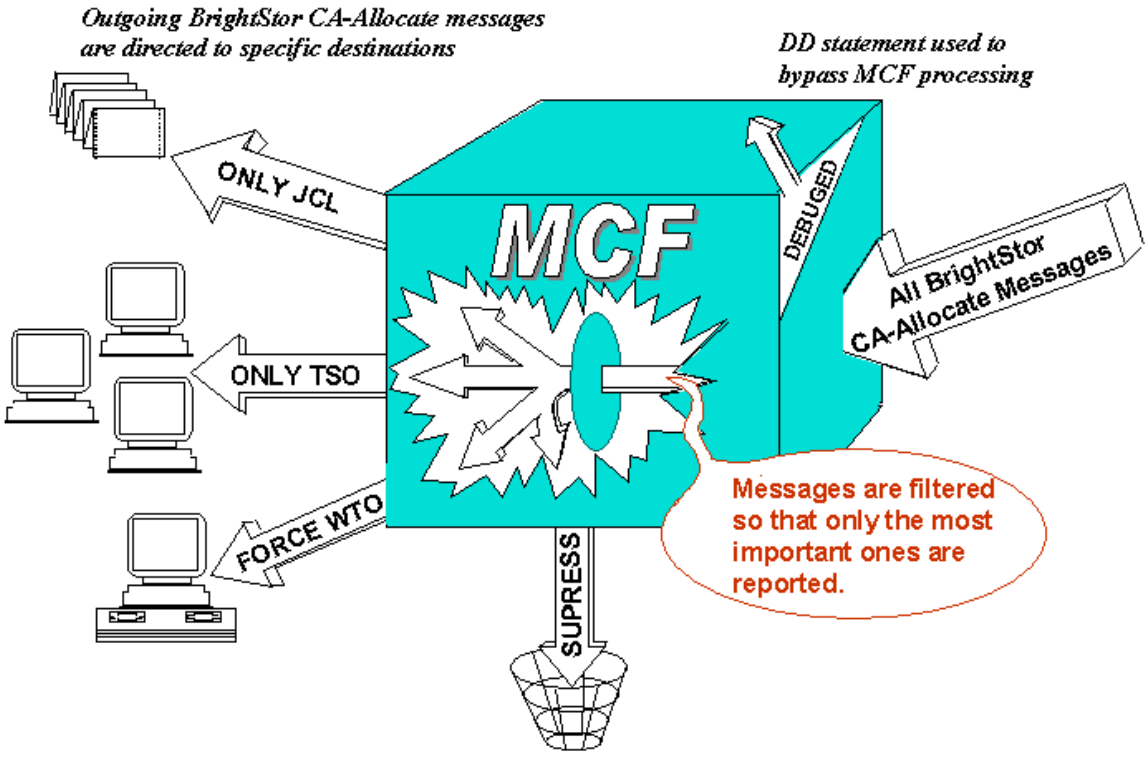
Appendix

D Message Control Facility

BrightStor CA-Allocate issues messages at many different points to inform you of its actions. Some messages seem unnecessary or only appropriate for specific individuals responsible for storage management. The message control facility allows you to discard these messages or direct them to a specific user or MVS console routing code.

You direct the message control facility by creating a message preprocessing table using assembler macro specifications that you assemble and link with BrightStor CA-Allocate.

Diagram of the Message Control Facility



After implemented, the message control facility is active for all allocations. It can be temporarily deactivated in any jobstep by adding the VDSDIAGS DD statement or by allocating it to your TSO session. The VDSDIAGS DD statement in TSO or batch JCL causes all messages to be displayed normally. To use a different ddname for this purpose, specify the DEBUGDD= keyword on the SCANBEG macro.

Macro Definitions

The message processing specifications are defined by three macros, SCANBEG, SCANSTR, and SCANEND.

Except for the optional DEBUGDD= keyword of SCANBEG, all specifications are defined by the SCANSTR macro.

SCANBEG is specified one time at the start of the member, and SCANEND is specified at the end. Descriptions of each macro follow:

```
SCANBEG DEBUGDD=VDSDIAGS
```

Begins the start of the table. This must be the first and only instance of this macro within the assembly. Its optional operand (DEBUGDD=) defines the ddname used to bypass the message processing facility. If DEBUGDD is not specified, 'VDSDIAGS' is assumed.

```
SCANSTR 'string',
OPTIONS=(SUPPRESS,ONLYJCL,ONLYTSO,FORCEWTO,WAIT,JESLOG,VANKEY,VANTLOG),
NOTIFY=userID,
ROUTCDE(n,n,n),
DESC=n
```

SCANSTR defines a search string used to select messages for direction. Any number of separate SCANSTR statements can be specified.

The message text of each message issued is compared to the search string. Messages matching the text specification are processed according to the parameters specified on the SCANSTR statement.

There is one positional operand and several keyword parameters as follows:

- The positional operand defines the string used to identify messages that the keyword parameters apply to.
- Patterns can be specified with the following characters
- '*' or '%'
- '*' meaning any number of preceding or trailing characters match
- '%' meaning any character in this position matches

Keyword Operands

OPTIONS=(...,...)

Any combination of:

SUPPRESS - Discard the message. It is not displayed.

ONLYJCL - Issue the message for batch jobs only. TSO users do not receive messages associated with this parameter.

ONLYTSO - Issue the message only when processing a TSO allocation request. Batch jobs do not receive messages associated with this parameter.

FORCEWTO - Write a copy of the message to the MVS operator console.

WAIT - Wait for messages issued to TSO to be placed in the terminal output buffer. If no buffers are available the program is placed into a wait state until buffers become available and the output line is placed in them. BrightStor CA-Allocate normally does not wait for messages.

JESLOG - Do not write the message to the operator's console. The message is written to the job log and appears only in the user's output.

VANTKEY - Send messages directly to the TSO userid of the BrightStor CA-Vantage user logged onto the SMS Plus GUI.

VANTLOG - Send the message to the BrightStor CA-Vantage log.

NOTIFY=

Send messages directly to the TSO USERID specified on this parameter.

ROUTCDE=

Write messages to the destination assigned to the route code specified on this parameter. Refer to the IBM Supervisor Services and Macro Instructions documentation for the WTO macro for more information about the ROUTCDE parameter.

DESC=

In addition to a ROUTCDE for directing messages to consoles, you may also provide descriptor codes. These codes will be provided to the system Write To Operator facility. Refer to the IBM Supervisor Services and Macro Instructions (WTO macro) for more information about proper values for the DESC parameter.

```
SCANEND
```

End the table; it must be the last statement in this program. The SCANEND macro has no operands.

Message Control Facility Example Specifications

Sample specifications for the message control facility are contained in the installation library in member VDSWTOI.

Assembling and Linking Message Control Specifications

The JCL to assembly and link using SMP/E is provided in member WTOSMP. Modify the JCL to reflect the appropriate zone (PARM) and release (FMID).

When run, the job links your Message Control Specifications with each component of BrightStor CA-Allocate. These specifications take effect with the next start-up of BrightStor CA-Allocate. A REFRESH does not cause these specifications to take effect! REFRESH only implements ASR and pool definition changes. You must stop and start BrightStor CA-Allocate for these specifications to take effect.

Appendix

E

Optional Maintenance

Some of our original BrightStor CA-Allocate users require non-standard, non-supported optional maintenance (Optional ZAPs). If your facility is currently using BrightStor CA-Allocate with any of these optional maintenance items applied, you need to apply the current version after normal installation. This maintenance is available in the Install Library as members PInnnnnn (where nnnnnn is an entry from the table below).

You can determine if any of the optional ZAPs are being used in your system by checking the Maintenance ID Numbers (MIDs) in table below with the MIDs that are printed out at the startup of the task or in the messages when the STATUS or REFRESH command is issued.

List of Optional Features

PI#	MID	Brief Description
130392	092	Replace bogus unit names with 'SYSALLDA' at '453' time even if no space specified. Without this PI, tape unit is assumed.
130535	126	Disable the exclusion of old data sets from "OLD Environment" processing if no unit is associated with the allocation request.
130740	106	Update the JFCB of an implied DSORG=PS allocation to an actual one. In certain circumstances, this allows the BLKSIZE to be updated.
601076	001	Excludes privately mounted volumes from BrightStor CA-Allocate's Non-VSAM Volume Selection Logic.
601084	002	Overrides BrightStor CA-Allocate's default replacement of bogus unit names to 'SYSALLDA'.
601143	003	Allows Volume-Level security checking to be done during BrightStor CA-Allocate's Non-VSAM Volume Selection Process.
601201	004	Installations using the ROSCOE Program Product require special handling for their allocations.
601414	008	Allows Volume-Level security checking to be done during BrightStor CA-Allocate's VSAM Volume Selection Process.
601502	009	Allows the TSO Userid to be retrieved into the &USER variable at sites that do not have a security system.

PI#	MID	Brief Description
601587	015	Put the original unit “back” if Volume Selection fails and FINVS=’N’.
601599	129	Cancel the allocation following an EXIT CODE(8) from SPACE or QUOTA Environments.
601613	025	Changes the PMR#1334 Routine’s allocation from “Specific” to Non-Specific.”
601624	044	Installations using the ASTEX Program Product require special handling for their allocations.
601626	128	Excludes privately mounted volumes from BrightStor CA-Allocate’s VSAM Volume Selection Logic.
601658	130	Installations running ACF2 running without a “SAF” Interface require special handling for their allocations.
601720	132	This optional PI has been superseded by the PLSOPT10 system parameter.
601735	066	Disable the STIMER in VDSST451.
601753	070	Exclude tape allocations from ALLOC Environment
601764	076	Use LOG=ASIS instead of LOG=NONE in VDSSECUR.
601765	077	Allow the use of the PI601765 DDname to disable the exclusion of jobs with DCB EXITs from EOV processing.
601769	081	Allow the use of the PI601769 DDname to disable the inclusion of jobs with DCB EXITs in EOV processing.

Appendix

F

Reports

BrightStor CA-Allocate provides a series of reports to help track the status of disk allocation on the system. Using the series of current, history, and trend reports, you can view disk space allocations system-wide and by storage group, volume, and esoteric unit name.

These reports help to monitor the results of the system's storage group definition. For example, if the reports show unbalanced allocations in one area, you can modify that storage group definition and run the report again. In this way, you can improve your storage group definitions, EDT configuration, and esoteric unit name groups.

Watching the pattern of allocations on volumes each day helps you schedule archive activities and preallocations, because the reports enable you to anticipate when heavy system loads occur.

Except for the Quota reports, described in “Reporting Options”, all the reports are written using the SAS system. The standard versions of the reports meet the needs of most storage administrators, but SAS allows quick and easy customization and expansion of the reporting capability. The original report-generating programs can serve as examples or templates for your new programs and, with the SAS manual, you should be able to quickly develop a program tailored to meet your requirements.

Samples of all the reports appear in this appendix.

The History Data Set

The master history data set holds one record of space allocation for each volume for each of the last 31 calendar days that the VTOC reader was executed. This data set is used by the History, Snapshot, and Trend Reports.

Each allocated space record contains the following fields:

VOLSER	The volume serial name.
USEDPCT	Percent of total volume space that is allocated.
ENTDATE	The numeric date and time that the associated current report was executed. This is used for comparison and sorting.
TOTSP	The total storage capacity of the volume (in MB).
USEDSP	The total allocated space on the volume (in MB).

The data set is maintained by the same job (called VDSVTOCS) that executes the VTOC reader. After the VTOC reader is completed, the SAS program reads the external files and computes the current records for the history data set. The current record's data is then appended to the current history data set.

When the appending step is completed, the program deletes any records that are older than 31 days.

Modifying the Default History Period

To keep a longer or a shorter history, edit the following statement in the VDSVTOCS program:

```
OLDDATE = HIGHDATE - 31;
```

By changing the number 31 in the above statement, you can easily change the number of records kept in the history data set. To keep a longer history period, increase the number. To keep a shorter history period, decrease the number.

JCL to Allocate the Reporting Data Sets

In order to run the reports successfully, two data sets must be allocated. These data sets are used for the history data and the data from the VTOC reader. The member VAMALLOC allocates these data sets for approximately 100 volumes.

The JCL must be customized as follows:

1. Supply an appropriate jobcard.
2. Set symbolic parameter Q to the desired high-level qualifier. Use the same high-level qualifier as for the installation of BrightStor CA-Allocate.
3. The supplied JCL allocates enough space for data on 100 volumes. If more or less than 100 volumes are reported on, adjust the two SPACE parameters proportionately.

JCL for Running the VTOC Reader

Member VDSVTOCS executes the VTOC reader load module and should be run daily. We recommend running the JCL during the least busy time on a system. The job should be run at approximately the same time each day.

The JCL must be customized as follows:

1. Supply an appropriate jobcard.
2. Change the high-level qualifiers if the default names are not being used.
3. The first time this job is run, a non zero completion code is received because the history file does not yet exist.

Run the job a second time to begin collecting the data.

JCL for Generating the Reports

The JCL needed to generate the SAS reports is provided in the Install library. This JCL can be copied and modified for each report.

General Guidelines for Modifying VDSRPTxx JCL

The guidelines for modifying VDSRPTxx JCL to generate the SAS reports are as follows:

1. Supply an appropriate jobcard.
2. Use the comments lines to supply the name and a brief description of each report.
3. Change the high-level qualifiers if the default names are not being used.
4. The //VDSINx DD supplies the data set name for the storage group definitions or the EDT on disk, depending on what report is running.

If the esoteric names are to be read from the disk EDT instead of from the incore EDT, its name must be supplied here. Normally, the incore EDT is read, and no JCL or program changes are required. However, if the EDT from disk is used, the SAS programs VDSRPTCE, VDSRPTHE, and VDSRPTTE must be modified. See the comments in these SAS programs for further details.

External Called Programs

Some of the SAS programs call external subroutines to read storage group definitions and the EDT configuration.

These load modules must be accessible to SAS. This can be accomplished by adding them to the SAS library, or by naming the load library on a STEPLIB statement in the JCL. This procedure is shown in the VDSRPTxx JCLs.

Managing the Report Output

Controlling What Volumes Are Reported

For the VDSRPTCV, VDSRPTHV, and VDSRPTTV reports, you can select the volumes to be included.

Before generating the report, the program checks which volumes are referenced by the storage group definitions. If a volume is not defined in a storage group, it is not included in the report even though its data is still recorded in the history data set. By pointing the JCL to a special storage group definition data set, you can limit the volumes included in the report.

Controlling What Storage Groups Are Reported

For the VDSRPTCP, VDSRPTHP, and VDSRPTTP reports, you can select the storage groups to be included.

Before generating the report, the program checks which volumes are referenced by the storage group definitions. If a volume is not defined in a storage group, it is not included in the report even though its data is still recorded in the history data set. By pointing the JCL to a special storage group definition data set, you can limit the storage groups included in the report.

Charting and Plotting Versus Printing

All the reports generate output by one of the two SAS procedures CHART or PLOT. These procedures present data in a graphic format.

To list the data before it is charted, use the SAS procedure PRINT. In the SAS programs for each of the history and trend reports, the appropriate SAS statements to print the data are “commented out” to reduce the number of printed pages produced. To change the program, remove the comment markers as described inside the SAS programs.

The History Reports

The history reports (VDSRPTDL,VDSRPTHV, VDSRPTHS, VDSRPTHP, and VAMRPTHE/VDSRPTHE) read all the data from the history data set.

Each report is described below.

VDSRPTDL

Detail Volume Usage Reports

These three reports provide detailed DASD usage statistics for individual volumes, volumes sorted by storage groups, and summary data for each storage group.

VDSRPTHV

History of Disk Space Allocated—by Volser

This report reads the history data set and uses PROC PLOT to report space allocated for each volume by date. It then generates a separate one-page plot for each volume showing the last 31 times the VDSRPTHV reader was executed and the percentage of space allocated each day.

VDSRPTHS

History of Disk Space Allocated—System-wide

This report totals the USEDSP and TOTSP variables in the history data set. Then the program calculates the used percentage from those two values and plots the result. It then generates a separate one-page plot for each volume showing the last 31 times the VDSRPTHS reader was executed and the percentage of space allocated each day.

VDSRPTHP

History of Disk Space Allocated—by Storage Group

Using the data in the history data set, this report gathers an associated storage group for each volser, then calculates and plots the percentage of space allocated for each storage group by date. It also generates a separate one-page plot for each volume showing the last 31 times the VDSRPTHP reader was executed and the percentage of space allocated each day.

VAMRPTHE

History of Disk Space Allocated—by Esoteric Unit Names

Using the data in the history data set, this report gathers an associated esoteric unit name for each volser, then calculates and plots the percentage of space allocated for each unit name by date. It also generates a separate one-page plot for each volume showing the last 31 days and the percentage of space allocated each day.

The Current Snapshot Reports

The current snapshot reports (VDSRPTCV, VDSRPTCS, VDSRPTCP, and VAMRPTCE/VDSRPTCE) use the most recent data from the history data set. The data is selected by finding the ENTDATE field of each record in the data set and keeping only those records with the highest value.

Each report is described below.

VDSRPTCV

Current Disk Space Allocated—by Volser

Using the most recent data from the history data set, the VDSRPTCV report determines the percentage of disk space allocated for each volume. After totaling the USEDSP and TOTSP values for each volume, it uses the PROC CHART to show the results.

VDSRPTCS

Current Disk Space Allocated—System-wide

Using the most recent data from the history data set, the VDSRPTCV report determines the percentage of disk space allocated on the entire system. After totaling the USEDSP and TOTSP values for each volume, it charts a total used percentage.

VDSRPTCP

Current Disk Space Allocated—by Storage Group

Using the most recent data from the history data set, the VDSRPTCP report gathers the associated storage group for each volser in the list, then totals and charts the percentage of space allocated by storage group.

VAMRPTCE

Current Disk Space Allocated—by Esoteric Unit Names

Using the most recent data from the history data set, the VDSRPTCE report gathers the associated esoteric unit names for each volser in the list, then totals and charts the percentage of space allocated by esoteric unit name.

The Trend Reports

The trend reports use the SAS procedure REG. The values predicted can be no more accurate than the USEDPCT and USEDSP values in the history data set. Therefore, the VTOC reader should be run consistently at the same time every day. Additionally, by extending the history period, you can increase the accuracy of the predicted value. For more details see the section Modifying the Default History Period in this appendix.

On the plotted output, the actual values are shown with an asterisk (*). The trend values (including the predicted values) are shown with a dollar sign ($). These are identified in the legend at the top of the plots.

The trend reports (VDSRPTTV, VDSRPTTS, VDSRPTTP, and VAMRPTTE/VDSRPTTE) predict how much disk space is allocated during the next 30 days. To change the prediction time period, change the number 30 in the following line in the appropriate SAS program:

```
DO X = 1 TO 30;
```

Each report is described below.

VDSRPTTV

Trend of Disk Space Allocated—by Volser

The VDSRPTTV report uses the history data set to determine the trend for the disk space allocated on each volser.

One plot is generated for each volume.

VDSRPTTS

Trend of Disk Space Allocated—System-wide

The VDSRPTTS report uses the history data set to determine the trend for the disk space allocated on the entire system. Only one plot is generated.

VDSRPTTP

Trend of Disk Space Allocated—by Storage Group

The VDSRPTTP report uses the history data set to determine the trend for the disk space allocated for each storage group. One plot is generated for each storage group.

VAMRPTTE

Trend of Disk Space Allocated—by Esoteric Unit Names

The VAMRPTTE/VDSRPTTE program uses the history data set to determine the trend for the disk space allocated for each esoteric unit name. One plot is generated for each esoteric unit name.

Sample Reports

The examples below are intended to give users the opportunity to become familiar with the functions, features, and results of the reports.

Volume Summary Detail Report

```
File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
--------------------------------------------------------------------------------------------------------------------------------
EDIT                        VDSRPTDL                                                          Columns 00006 00129
Command ==> _____________________________________________________________________________________________ Scroll ==> CSR
000119 SY9CAT   3380      84    1891    1593     298      416    6286        10        271        4065      47476       15
000120 SY9PG1   3380     100    1891    1891       0        0       9         1          0           9      47476       15
000121                                                CA-ALLOCATE                        11:06 Monday, October 15, 2001
000122                                     VOLUME SUMMARY DETAIL REPORT
000123
000124         DEVICE  PERCENT  TOTAL   USED    FREE     FREE    FREE   FREESPACE   LAREGEST    LARGEST    TRACK     TRACKS
000125 VOLSER   TYPE      USED  MBYTES  MBYTES  MBYTES    CYLS    TRKS    EXTENTS   EXTENT/CYL EXTENT/TRK   SIZE     PER CYL
000126 ------  ------  -------  ------  ------  ------    ----    ----   ---------  ---------- ----------   -----     -------
000127
000128 SY9PG2   3380     100    1891    1890       0        0      19         3          0           9      47476       15
000129 SY9PG3   3380     100     630     630       0        0       4         1          0           4      47476       15
000130 SY9PG4   3380     100     630     630       0        0       4         1          0           4      47476       15
000131 SY9SP1   3380     100    1891    1890       0        0      14         1          0          14      47476       15
000132 SY9SY1   3380      99    1891    1872      19       16     402        37          5          86      47476       15
000133 SY9SY1   3380      99    1891    1872      19       16     402        37          5          86      47476       15
000134 SY9SY2   3380      90    1891    1693     197      197    4164       306         75        1126      47476       15
000135 SY9SY2   3380      90    1891    1693     197      197    4164       306         75        1126      47476       15
000136 SY9SY3   3390      27    1892     511    1380     1624   24367         1       1624       24367      56664       15
000137 SY9SY3   3390      27    1892     511    1380     1624   24367         1       1624       24367      56664       15
000138 TRAINS   3390      84     630     529     100      139    2118         5        106        1600      47476       15
000139 390390   3380      77    1891    1456     435      611    9166         3        583        8745      47476       15
000140
000141
```

Storage Group Detail Report

```
File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------------------------------------------------
EDIT                       VDSRPTDL                                                      Columns 00010 00133
Command ==> ____________________________________________________________________________ Scroll ==> CSR
000194                                      CA-ALLOCATE                          11:06 Monday, October 15, 2001
000195                                  STORAGE GROUP DETAIL REPORT
000196
000197 ---------------------------------- STORAGE#GROUP#-------=DMSVOL-----------------------------------------
000198
000199               PERCENT     TOTAL     USED     FREE     FREE     FREE     FREESPACE     LAREGEST      LARGEST
000200     VOLSER      USED     MBYTES    MBYTES   MBYTES   CYLS     TRKS      EXTENTS     EXTENT/CYL   EXTENT/TRK
000201     ------     -------   ------    ------   ------   ----     ----     ---------    ----------   ----------
000202
000203     DMSE01        61       1260      773      487      624    10266          83           66         1004
000204     DMSK01        57       1891     1087      804     1046    16945         148          140         2100
000205     DMST01        57        630      359      270      360     5702          47          112         1689
000206     DMST02        48        630      301      328      437     6928          42           76         1151
000207     DMS901        77       1892     1458      433      371     7658         319           36          541
000208     DMS902        60       1892     1141      751      778    13260         215           45          684
000209     DMS903        55       1892     1048      844      847    14901         263           70         1050
000210                            ------   ------   ------    ----     ----     ---------
000211     STORGRP               10087     6167     3917     4463    75660        1117
000212
000213                                        N = 7
000214
```

Storage Group Summary Report

```
File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------------------------------------------------
EDIT                       VDSRPTCP                                                                  Columns
00008 00131
Command ==> ____________________________________________________________________________ Scroll ==> CSR
000531                                                          CA-ALLOCATE              11:06 Monday, October
15, 2001
000532                                                  STORAGE GROUP SUMMARY REPORT
000533
000534 STORAGE      VOLUMES    PERCENT     TOTAL     USED       FREE     FREE      FREE  FREESPACE   LAREGEST    LARGEST
000535 GROUP        IN GROUP     USED      MBYTES    MBYTES     MBYTES   CYLS      TRKS  EXTENTS     EXTENT/CYL  EXTENT/TRK
000536 --------     --------   ------     ------    ------     ------   ----      ----  ---------   ----------  ----------
000537
000538 ABOVEUCB         1          84       1892      1602       289      333      5117       16         149         2248
000539 DASD80           9          72      10083      7289      2790     3589     58859      789         882        13243
000540 DASD90           7          63      13243      8395      4845     5396     88747     1009        1624        24367
000541 DMSVOL           7          61      10087      6167      3917     4463     75660     1117         140         2100
000542 EVEN             3          89       3781      3367       412      308      8719      778           8          131
000543 FAKE90           3          64       5676      3647      2028     1996     35819      797          70         1050
000544 FRGVOL          11          65      19550     12817      6727     7478    120976     1241        1952        29281
000545 MVSVOL          22          77      34042     26247      7784     8526    148044     3406         170         2550
000546 ODD              4          90       6933      6288       644      498     13597     1195           9          149
000547 ONEREAL          1          90       1892      1717       174      202      3082       13          57          855
000548 QAVOLS           3          91       3152      2877       275      175      5058      300           6          100
000549 REAL90           5          71       9460      6751      2706     3155     47809      113         170         2550
000550 SAMVOL           1          50       1891       961       930     1295     19595       20        1252        18780
000551 SSLVOL           7          50      10714      9655      1056      806     22316     1973           9          149
000552 SY9VOL          11          85      19227     16388      2835     3489     53876      376        1624        24367
000553 VOL801           1          89       1891      1698       193      151      4074      343           9          149
```

Current Disk Space Allocated-Storage Group

```
File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------------------------------------------------------
EDIT                     VDSRPTCP                                                                  Columns 00006 00129
Command ==> ____________________________________________________________________________________________ Scroll ==> CSR
000001                                       CA-ALLOCATE                                  11:06 Monday, October 15,
2001
000002                                   CURRENT DISK SPACE ALLOCATED
000003
000004 STORAGE GROUP                                                                                      PERCENT
USED
000005                                                                                                         Sum
000006           |
000007 ABOVEUCB  |*************************************************************************************      84.6723
000008           |
000009 DASD80    |*********************************************************************                      72.2900
000010           |
000011 DASD90    |************************************************************                               63.3920
000012           |
000013 SAMVOL    |**************************************************                                         50.8197
000014           |
000015 VOL902    |************************************************************                               60.3066
000016           |
000017 VOL903    |*****************************************************                                      55.3911
000018           |
000019           |----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+
000020                5    10   15   20   25   30   35   40   45   50   55   60   65   70   75   80   85   90   95  100
000021
000022                                                 PERCENT USED
****** ***************************************************** Bottom of Data ********************************************************
```

Current Disk Space Allocated-Unit Name

```
File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------------------------------------------------------
EDIT                     VDSRPTCE                                                                  Columns 00006 00129
Command ==> ____________________________________________________________________________________________ Scroll ==> CSR
000001                                       CA-ALLOCATE                                  11:06 Monday, October 15,
2001
000002                                   CURRENT DISK SPACE ALLOCATED
000003
000004 UNIT NAME                                                                                          PERCENT
USED
000005                                                                                                         Sum
000006           |
000007 DISK      |*************************************************************************                 76.6431
000008           |
000009 SYSALLDA  |*************************************************************************                 76.6431
000010           |
000011 SYSDA     |**************************************************************************                77.7006
000012           |
000013 VIO       |*************************************************************************************      83.4656
000014           |
000015 3380      |************************************************************************************     81.8592
000016           |
000017 3390      |*****************************************************************                          68.3489
000018           |
000019           |----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+
000020                5    10   15   20   25   30   35   40   45   50   55   60   65   70   75   80   85   90   95  100
000021
000022                                                 PERCENT USED
```

Current Disk Space Allocated-System-Wide

```
File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------------------------------------------------------
EDIT                       VDSRPTCE                                                          Columns 00006 00129
Command ==> ____________________________________________________________________________________ Scroll ==> CSR
                                                 CA-ALLOCATE                        11:06 Monday, October 15,
2001
                                              CURRENT DISK SPACE ALLOCATED
                                                      SYSTEM WIDE
          DATE                                                                                          PERCENT
USED
                                                                                                                Sum
                |
09APR  96:08:23 |*************************************************************************                    74.34799
                |----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+
                     5    10   15   20   25   30   35   40   45   50   55   60   65   70   75   80   85   90   95  100

                                                       PERCENT USED
************************************************************ Bottom of Data ********************************************************
```

History of Disk Space Allocated-Storage Group=ABOVEUCB

```
File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------------------------------------------------------
EDIT                       VDSRPTHP                                                          Columns 00006 00129
Command ==> ____________________________________________________________________________________ Scroll ==> CSR
000001                                           CA-ALLOCATE                        11:06 Monday, October 15,
2001
000002                                        HISTORY OF DISK SPACE ALLOCATED
000003
000004 ------------------------------------------------ STORAGE GROUP=ABOVEUCB ------------------------------------------------
000005
000006           |
000007           |
000008       100 +-------------------------------------------------------------------------------------------------------------
000009           |                                  *  *  *  *                  *  *  *  *  *
000010           |                               *           *  *  *  *  *  *
000011        90 +                            *
000012           |                           *                                                 *  *  *  *  *  *  *
000013           |                                                                                                  *
000014        80 +-------------------------------------------------------------------------------------------------------------
000015           |
000016           |                  *
000017   P    70 +  *  *  *  *  *    *  *  *
000018   E       |
000019   R       |
000020   C    60 +-------------------------------------------------------------------------------------------------------------
000021   E       |
000022   N       |
000023   T    50 +
```

Appendix

G

ISPF Interface

Note: The ISPF Interface is not supported after release 5.3.

The ISPF interface provides you with a simple method for defining VSAM clusters (KSDS, ESDS, RRDS) and alternate indexes (AIXs). You select a define function from a menu listing the various define options, then the appropriate ISPF screen is displayed. This appendix shows examples of how to install and implement the various ISPF screens available.

After you supply definition information, BrightStor CA-Allocate creates IDCAMS define statements that are passed dynamically to IDCAMS. There is no need to wait for a batch job to run—the define is executed immediately and the result is returned to you.

Note that BrightStor CA-Allocate standards enforcement capability is invoked when IDCAMS issues the define to catalog management, not during the ISPF interface prompts. Therefore, define requests that are inconsistent with your installation's standards are not detected until the actual define occurs (SVC 26 is issued). This can create the illusion that the define is accepted as entered, when in reality BrightStor CA-Allocate may fail, modify or redirect the define in accordance with the processing options that have been supplied, in the ASR routines.

Note also that an ISPF screen is available to permit the user to save commands that were generated in a partitioned or sequential data set for later execution.

BrightStor CA-Allocate also provides a facility that lists catalog information for a cluster. This option can be invoked directly from the primary option menu or after a define is completed.

ISPF Implementation and Installation

The ISPF interface is installed on an IBM-supplied Primary option menu. All dialog management panels have associated HELP text panels. If at any time additional information is needed about one of the panels, press the <HELP> PF key. The HELP text can be browsed. The <END> key takes you back to the panel where you were processing originally.

The ISPF support is shipped on the distribution tape as two separate files:

- SAMPNL0 — ISPF Dialog Manager Panel Library. This library contains the panel definitions for all dialog manager panels and contains all HELP facility panels relating to the function and menu panels.
- SMMSG0 — ISPF Dialog Manager Message Library. This library contains messages to be displayed by the dialog manager when the application detects an abnormal condition.

The process of implementing the dialog manager interface consists of the following three steps:

1. Loading the supplied libraries from tape, already done during SMP/E installation.
2. Concatenating the libraries to the existing ISPF dialog management libraries.
3. Adding “BrightStor CA-Allocate“ to the appropriate ISPF option menu(s).

Step 1. Loading the ISPF Libraries from Tape

This step was already done during the SMP/E installation.

Step 2. Connecting the ISPF Libraries to Your System

Before invoking the ISPF support, follow the steps below to allocate the dialog manager libraries TSO sessions.

1. Determine how your installation allocates the data sets referred to by the DDNAMES below to the TSO session before invoking the ISPF facility. They can be allocated by including them in the TSO LOGON JCL procedure or through a CLIST that is executed sometime before ISPF is invoked.

   DDNAME = ISPPLIB panel library

   DDNAME = ISPMLIB message library

2. Update the allocation of these libraries to concatenate the BrightStor CA-Allocate ISPF libraries after them. The names of the unloaded libraries correspond to the DDNAMES to which they should be concatenated.

   Be sure to check the block sizes of the libraries being concatenated. The block size of the first library must be greater than or equal to that of the BrightStor CA-Allocate ISPF library or errors occurs. This problem can be corrected by either reblocking the libraries or simply putting DCB=BLKSIZE=nnnnn on the dd statement for the first data set of the concatenation, where nnnnn corresponds to at least the largest of the block sizes in the concatenation.

   For example, here is the JCL before libraries are concatenated:

   ```
   //ISPPLIB   DD  DISP=SHR,DSN=ISP.R1MO.AVDSPLIB
   //ISPMLIB   DD  DISP=SHR,DSN=ISP.R1MO.AVDSMLIB
   ```

   Here is the JCL after the libraries are concatenated:

   ```
   //ISPPLIB   DD  DISP=SHR,DSN=ISP.R1MO.ISPPLIB
   //          DD  DISP=SHR,DSN=VAM.SAMPNL0
   //ISPMLIB   DD  DISP=SHR,DSN=ISP.R1MO.ISPMLIB
   //          DD  DISP=SHR,DSN=VAM.SAMMSG0
   ```

3. The load library that was specified by SAMLINK contains the ADSVSPF1, ADSVSPF2, and ADSVSPF3 after SMP/E installation.

Step 3. Incorporating the ISPF Interface with the ISPF Options Menu

To install the interface into the standard IBM options menu, do the following:

1. A minimum of one modification to IBM-distributed ISPF panels is required to implement the ISPF support. This modification allows the ISPF support to be accessed from the ISPF primary option menu. To do this, at least one option must be added to the primary option menu. ISPF/PDF users can choose to make this change to the master application menu (see below).

   First, locate the installation's ISPF panel library and find the primary option menu. It may be in member ISP@PRIM or member ISR@PRIM, depending on the level of ISPF. Find an existing line in the panel that looks like the following sample:

   ```
   % 7 +SUPPORT - TEST DIALOG OR CONVERT MENU/MESSAGE FORMATS
   ```

   If the last entry looks like the sample, the next option is 8. If not, subsequent entries may appear after this one.

   In which case, select the next sequential option number or letter available.

   Implementation can be at one or two levels, depending on what options are to be made available to the end users. The high level accesses a simple menu which lists all of the options provided. This panel's ID is VAM012. If users are to have access to all functions, enter the following line (with the appropriate option number) after the last option which was defined:

   ```
   % 8 +VAM - PERFORM VSAM ALLOCATION MANAGER FUNCTIONS
   ```

2. Find an existing line in the panel that looks like the following sample (which is in the ')PROC' section of the panel):

   ```
   7,'PANEL(ISPQTAC) NEWPOOL'
   ```

   Following this line (or the last one being used), insert the line below:

   ```
   8,'PANEL(VAM012)'
   ```

   Note: Use the same option number as is used above.

   If users are not to have access to all functions, enter only those specific options that are to be made available. Follow the same general modification scheme outlined above, but enter the specific option(s):

   ```
   % 8 +DEFINE   -  DEFINE VSAM CLUSTER USING VAM
   % 9 +DEFINE   -  DEFINE VSAM CLUSTER USING VAM (ABBREVIATED)
   % a +LISTCAT  -  DISPLAY CATALOG INFORMATION FOR CLUSTER
   ```

3. In the ')PROC' section of the panel, add the corresponding entries:

```
8,'PGM(ADSVSPF1)
9,'PGM(ADSVSPF2)
A,'PGM(ADSVSPF3)
```

ISPF / PDF users who also uses the master application menu ISR@MSTR, may make the above change(s) to this panel instead of the primary option menu ISR@PRIM. If so, the modification(s) in the ')PROC' section of the panel should follow this general pattern:

```
8,'PGM(VSPF1) NEWAPPL(ISR)'
```

This procedure makes BrightStor CA-Allocate's ISPF Feature a primary application like PDF and sets the application profile which is used for the profile information. To have it use the same application profile information as PDF, make sure that the value of the NEWAPPL parameter of the line is the same as the value of the NEWAPPL parameter in the line that looks something like this:

```
1,'PANEL(ISR@PRIM) NEWAPPL(ISR)'
```

At this point, the ISPF interface is fully functional. Any users logged onto ISPF during the installation process need to logoff their TSO session and then logon again. This allows ISPF to access all new libraries and new panels. To test, choose option 8 (or whatever option you selected for BrightStor CA-Allocate) on the primary option menu. From this point on, use the HELP text (the PF key) to proceed through ISPF panel usage.

Pattern Matching with the ISPF Interface

This section describes pattern matching as used with the ISPF Interface. This pattern matching feature is not the same as the pattern matching used for specifying Allocation Selection Routine constants that is described in the section **Patterns** in the chapter "**Implementation**."

When properly used, the ISPF Interface pattern matching capability can reduce dramatically the time required to enter commands. The data set name is a common parameter using pattern matching; it is used as an example in this section. Other parameters using pattern matching include User IDs and job and program names.

Either full data set names or data set name patterns to be entered in any DSNAMES= parameter. A pattern name consists of the usual alpha-numeric and national characters allowed in a data set name, but includes the following as well:

- an asterisk (*)
- a question mark (?)
- a slash (/)
- an exclamation point (!)

The asterisk (*) can be used to represent any variable index level or simple name.

DSN=*	selects all single-level data set names.
DSN=*.*	selects all two-level data set names.
DSN=A.*.PROD	selects all three-level names that have "A" as their first index and "PROD" as the simple name, but any second-level index.
DSN=A*.PROD	selects all data sets having a first-level index beginning with the letter "A" followed by any other characters, and a second-level index equal to the value "PROD."

The question mark (?) can be used to represent any variable character within an index level or simple name. Multiple occurrences can be used within each level or simple name.

DSN=?	selects all single-character data set names.
DSN=A.TEST??	selects all two-level data set names with a first-level index of "A" and a simple name six characters in length, the first four of which must be "TEST" (TEST01, TEST02, TEST1A, and so on).

The slash (/) can be used to represent any variable character from that position to the end of the name. The portion of the name that precedes the slash is referred to as a prefix name.

DSN=A/	selects all data sets that begin with the character "A."
DSN=A.TEST/	selects all data sets that begin with the character string "A.TEST" with any characters following.
DSN=A.*.C?./	selects all data sets that begin with an index of "A" followed by any second-level index, a two-character third-level index starting with "C" and any following string.

The exclamation point (!) can be used to represent any variable character up to the character string following the exclamation point. It defines the beginning of a character string (terminated by the next pattern character or the end of data) that can be found anywhere within the name.

DSN=!TEST	selects all data sets that contain "TEST" somewhere in the name.
DSN=A?.!DEPT2	selects all data sets that have a two-character first-level index that starts with an "A" and contains "DEPT2" somewhere in the remainder of the name.
DSN=!TEST!VSAMFILE	selects all data sets that contain "TEST" somewhere in the name and "VSAMFILE" somewhere following the "TEST."

ISPF Panels

This is the primary option panel for the BrightStor CA-Allocate ISPF interface. Enter the number of the function you want.

FIGURE G-1 Primary Option Menu

```
VAN012 ----------------------- PRIMARY OPTION MENU ----------------- CA-ALLOCATE
OPTION ===> _

    0  FULL DEFINE - DEFINE A VSAM CLUSTER
    1  ABBR DEFINE - DEFINE A VSAM CLUSTER (ABBREVIATED)
    2  LISTCAT     - DISPLAY CATALOG INFORMATION FOR CLUSTER









   ENTER END COMMAND TO TERMINATE CA-ALLOCATE FUNCTIONS.
```

This is the head panel displayed when defining a cluster using the long version. Most of the fields are self-explanatory to those who know VSAM definition processing. If a model cluster is specified that has a pattern-matching character in it (as in this example), a catalog locate is issued and a scrollable list of data set names matching the pattern is presented for the user to make the selection.

FIGURE G-2 Cluster Definition Panel

```
VAM001 ---------------------- CLUSTER DEFINITION ----------------- CA-ALLOCATE
COMMAND ===> _

CLUSTER NAME  ===> october.workship.ksds
CLUSTER TYPE  ===> k       (KSDS, ESDS, RRDS,, LINEAR, OR AIX)
CLUSTER NAME  ===> october.workship.ksds
    OWNER ID  ===> ISPDLM1
CATALOG ==>                                                  PASSWORD ===>
-------------------------------------------------------------------------------
ALLOCATION DATA:     TYPE ===> C        (CYL,TRK, OR REC)
         PRIMARY QUANTITY ===> 2           SECONDARY ===> 1
-------------------------------------------------------------------------------
RECORD SIZE:     AVERAGE ===> 80             MAXIMUM ===>
-------------------------------------------------------------------------------
ENTER THE SPECIFIC VOLUME(S) DESIRED BELOW (UP TO 10)
===> ssl
ALLOCATE THE VOLUMES IN THE ORDER SHOWN? ===>      (YES/NO)
-------------------------------------------------------------------------------
IF MODELING DESIRED, GIE THE MODEL'S DATASET NAME AND PASSWORD (IF REQUIRED):
MODEL ===> 'ispsjj1.test./'                                  PASSWORD ===>
IF NEEDED, ENTER THE CATALOG NAME AND ITS PASWORD) WHERE MODEL CAN BE FOUND:
CATALOG ===>                                                 PASSWORD ===>
DISPLAY MODEL'S ATTRIBUTES ON SUCCEEDING PANELS? ===>        (YES/NO)
```

The next panel to be displayed depends on the cluster type being defined (extra information is required if an alternate index or KSDS is being defined).

Note: The data entered for variables is normally retained on the screen for multiple define requests. To clear the screen of input data, enter either "RESET" or "CLEAR" on the command line and press <ENTER>.

Here the scrollable list that is presented when a pattern data set name is supplied in the model parameter. The user can scroll the list and place an “S” next to the data set name that he or she wants to use. This name is carried over into succeeding panels where applicable.

FIGURE G-3 Catalog Data Set List Panel

```
VAM009 -------------------- CATALOG DATA SET LIST -----------     Row 1 to 14 of
14
COMMAND INPUT ===>                                              SCROLL ===>  CSR
-------------------------------------------------------------------------------

SELECT       TYPE      DSNAME
------    ---------    -------------------------
          CLUSTER      ISPSJJ1.TEST.ESDS1
          CLUSTER      ISPSJJ1.TEST.KSDS1
          CLUSTER      ISPSJJ1.TEST.KSDS2
  s       CLUSTER      ISPSJJ1.TEST.KSDS3
          CLUSTER      ISPSJJ1.TEST.LDS1
          CLUSTER      ISPSJJ1.TEST.RRDS1
          CLUSTER      ISPSJJ1.TEST.VSAM11
          CLUSTER      ISPSJJ1.TEST.VSAM12
          CLUSTER      ISPSJJ1.TEST.VSAM13
          CLUSTER      ISPSJJ1.TEST.VSAM5
          CLUSTER      ISPSJJ1.TEST.VSAM6
          CLUSTER      ISPSJJ1.TEST.VSAM7
          CLUSTER      ISPSJJ1.TEST.VSAM70
          CLUSTER      ISPSJJ1.TEST.VSAM71
------------------------------- Bottom of data --------------------------------
```

This panel is displayed whenever a KSDS or alternate index is being defined. It allows the user to specify information about the format of the keys in the data set as well as options regarding the format of the index.

FIGURE G-4 KSDS Alternate Index Information Panel

```
VAM001 ----------------------- INDEXING INFORMATION ----------------- CA-ALLOCATE
COMMAND ===>

CLUSTER NAME:          .OCTOBER.WORKSHOP.KSDS                    TYPE:KSDS
-------------------------------------------------------------------------------
             KEY INFORMATION: LENGTH ===> 10      OFFSET ===> 3

            PERCENTAGE OF FREESPACE PER CONTROL INTERVAL ===> 20
   PERCENTAGE OF FREE CONTROL INTERVALS PER CONTROL AREA ===> 1
-------------------------------------------------------------------------------
PLEASE ANSWER YES OR NO TO THE FOLLOWING QUESTIONS:

    IMBED  ===> Y     PLACE THE SEQUENCE SET OF THE INDEX WITH THE DATA?
REPLICATE  ===>       PUT EACH INDEX RECORD ON ITS OWN TRACK, AND REPLICATE IT?
```

The amounts of imbedded free space that are to be reserved in the data component of the cluster can also be specified.

This panel is displayed for all types of clusters. The user can code the expiration date in any format desired, and BrightStor CA-Allocate converts that into the format required by IDCAMS.

FIGURE G-5 Cluster Definition Panel

```
VAM002 ----------------------- CLUSTER DEFINITION ----------------- CA-ALLOCATE
COMMAND ===>

CLUSTER NAME:          .OCTOBER.WORKSHOP.KSDS                   TYPE:KSDS
--------------------------------------------------------------------------------
RETENTION PERIOD: EXPLICIT DATE ===> 10          <OR> NUMBER OF DAYS ===>
SHARE OPTIONS: CROSS REGION (1,2,3,4) ===> 1
               CROSS SYSTEM (3 OR 4)  ===> 3
CONTROL INTERVAL SIZE ===>          BUFFERSPACE ===> 50000
--------------------------------------------------------------------------------

PLEASE ANSWER YES OR NO TO THE FOLLOWING QUESTIONS:

 COMP DETAIL  ===> y  REPLY YES TO DISPLAY ADDITIONAL PANELS FOR DATA/INDEX
   PASSWORDS  ===> y  REPLY YES TO DISPLAY PANEL WITH PASSWORD PARAMETERS

       REUSE  ===> n  CAN CLUSTER BE WRITTEN OVER BASED ON OPEN OPTION?
WRITE CHECKS  ===> y  READ EACH RECORD AFTER IT IS WRITTEN FOR ADATA CHECKS?
     SPANNED  ===> n  CAN A DATA RECORD CROSS CONTROL INTERVAL BOUNDARIES?
     RECOVER  ===>    SHOULD THE CLUSTER BE PREFORMATTED DURING INITIAL LOAD?
       ERASE  ===>    OVERWRITE THE CLUSTER WITH BINARY ZEROES WHEN DELETED?
      UNIQUE  ===>    DEFINE THE CLUSTER IN ITS OWN DATA SPACE (NON-ICF ONLY)?
```

If specific definition parameters are desired at the data or index levels, the user replies “yes” to the COMP DETAIL prompt on the screen. This leads to another set of panels for entering the data and index information. If password information is desired, specify “yes” to the PASSWORDS prompt on the screen. This displays a special panel used for inputting password information. The remaining questions on the screen can be answered or simply left blank to assume the default that IDCAMS assigns.

Here is the panel used for inputting password information. Again, this panel is displayed only if the user replied “yes” to the PASSWORDS prompt on the previous screen.

FIGURE G-6 Password Specification Panel

```
VAM003 -------------------- PASSWORD SPECIFICATION ----------------- CA-ALLOCATE
COMMAND ===>

CLUSTER NAME:          .OCTOBER.WORKSHOP.KSDS                    TYPE:KSDS
--------------------------------------------------------------------------------
         NUMBER OF ATTEMPTS  ===> 3
CODE FOR OPERATOR PROMPTING  ===> wrksp
--------------------------------------------------------------------------------
SUPPLY THE PASSWORDS YOU WANT FOR THIS COMPONENT:
            MASTER PASSWORD  ===>
           CONTROL PASSWORD  ===>
            UPDATE PASSWORD  ===>
              READ PASSWORD  ===>
--------------------------------------------------------------------------------

ENTER SPECIAL AUTHOORIZATION INFORMATION BELOW:
               PROGRAM NAME  ===>
         AUTHORIZATION DATA  ===>
```

Password input fields are marked as non-displayable for security reasons.

This is the panel that is displayed when the user replied “yes” to the component detail prompt on panel VAM002. The panel is used to enter specific information about the data component.

FIGURE G-7 Data Component Detail Panel

```
VAM001 --------------------- DATA COMPONENET DETAIL ----------------- CA-ALLOCATE
COMMAND ===>

CLUSTER NAME:           .OCTOBER.WORKSHOP.KSDS                   TYPE:KSDS
--------------------------------------------------------------------------------
             DATA COMPONENT NAME  ===> ‘          .OCTOBER.WORKSHOP.KSDS
            CONTROL INTERVAL SIZE ===> 4096
--------------------------------------------------------------------------------
ALLOCATION DATA:             TYPE  ===> r          (CYL, TRK, OR REC)
                PRIMARY QUANTITY  ===> 200          SECONDARY ===>
--------------------------------------------------------------------------------
ENTER THE VOLUME(S) DESIRED FOR THIS COMPONENET BELOW (UP TO 10):
===> ss1807
ALLOCATE THE VOLUMES IN THE ORDER SHOWN? ===>       (YES/NO)
```

In this example, the volume “SSL807” has been requested for the data component, along with a specific control interval size and allocation quantity.

Here is the panel used to enter specific parameters for the index component of the cluster.

FIGURE G-8 Index Component Detail Panel

```
VAM0014 -------------------- INDEX COMPONENET DETAIL ---------------- CA-ALLOCATE
COMMAND ===>

CLUSTER NAME:          .OCTOBER.WORKSHOP.KSDS                   TYPE:KSDS
--------------------------------------------------------------------------------
             INDEX COMPONENT NAME ===> '         .OCTOBER.WORKSHOP.KSDS
            CONTROL INTERVAL SIZE ===> 1024
--------------------------------------------------------------------------------
ALLOCATION DATA:             TYPE  ===> +          (CYL, TRK, OR REC)
                  PRIMARY QUANTITY  ===> 2            SECONDARY ===>
--------------------------------------------------------------------------------
ENTER THE VOLUME(S) DESIRED FOR THIS COMPONENET BELOW (UP TO 10):
===> ss1808
ALLOCATE THE VOLUMES IN THE ORDER SHOWN? ===>       (YES/NO)
```

In this example, a different volume (SSL808) is being requested for the index component.

The message panel appears after a define was successful. The messages that are returned by IDCAMS are shown on the bottom of the screen. The physical volumes that BrightStor CA-Allocate selected from the requested pools are shown in the messages.

FIGURE G-9 IDCAMS DEFINE Definition Results Panel

```
VAM005 ----------------------- DEFINITION RESULTS ------------------ CA-ALLOCATE
COMMAND ===>

CLUSTER NAME:          .OCTOBER.WORKSHOP.KSDS                   TYPE:KSDS
--------------------------------------------------------------------------------
DISPLAY IDCAMS COMMANDS? ===> y     (YES/NO)
   SAVE IDCAMS COMMANDS? ===> y     (YES/NO)
   DISPLAY CATALOG INFO? ===> y     (YES/NO)
--------------------------------------------------------------------------------
IDCAMS MESSAGES:                                                MSG TOTAL: 2

IDC0508I DATA ALLOCATION STATUS FOR VOLUME SY9SY1 IS 0
IDC0509I INDEX ALLOCATION STATUS FOR VOLUME SY9SY1 IS 0
```

In this example, a display of the IDCAMS statements, a save of the commands, and a catalog listing of the cluster was requested.

The scrollable list of commands that were passed to IDCAMS for the definition then appears. This is the exact list that can be saved into a user-specified data set.

FIGURE G-10 IDCAMS Control Statements Panel

```
VAM006 ----------------------IDCAMS CONTROL STATEMENTS ------- Row 1 to 19 of 27
COMMAND ===>

                 LIST OF IDCAMS CONTROL STATEMENTS
----*----1----*----2----*----3----*----4----*----5----*----6----*----7----*----
 DEFINE CLUSTER           -
    (NAME (‘     .OCTOBER.WORKSHOP.KSDS’)          INDEXED         -
    OWNDER(ISPDLM1 ) –
    MODEL(‘SYSS4.OMII.TB’-
    ) –
    SHR(2 3) –
    NORWUSE  -
    NOWRITECHECK –
    NONSPANNED   -
    RECOVER –
    NOERASE-
    FREESPACE(0   0  ) –
    KEYS(128  0      ) –
    IMBED  -
    REPLICATE  –
    ATTEMPTS(3  ) –
    CODE(WRKSP   ) –
     ) DATA ( -
     NAME (‘     .OCTOBER.WORKSHOP.KSDS.DATA’)                              -
```

This is the panel that is displayed when users want to save IDCAMS control statements for later use. It is possible to directly edit the data set where the statements are saved by replying “yes” to the edit data set prompt.

FIGURE G-11 Save IDCAMS Command Statement Panel

```
VAM007 --------- SPECIFY DSNAME TO SAVE IDCAMS COMMANDS IN --------- CA-ALLOCATE
COMMAND ===>

CLUSTER NAME:          .OCTOBER.WORKSHOP.KSDS                    TYPE:KSDS
--------------------------------------------------------------------------------

DATA SET NAME ===>           .util.cntl
  MEMBER NAME ===> define

--------------------------------------------------------------------------------
EDIT DATASET? (YES/NO) ===>
```

Here is another example of the primary screen for a full define.

FIGURE G-12 IDCAMS DEFINE of an Alternate Index Panel

```
VAM001 ----------------------- CLUSTER DEFINITION ----------------- CA-ALLOCATE
COMMAND ===>

CLUSTER NAME ===> OCTOBER.WORKSHOP
CLUSTER TYPE ===> a   (KSDS, ESDS, RRDS, LINEAR, OR AIX)
    OWNER ID ===> ISPDLM1
CATALOG ==>                                                 PASSWORD ==>
--------------------------------------------------------------------------------
ALLOCATION DATA:     TYPE ===> t      (CYL, TRK, OR REC)
        PRIMARY QUANTITY ===> 15       SECONDARY ===> 5
--------------------------------------------------------------------------------
RECORD SIZE:       AVERAGE ===> 8000      MAXIMUM ===> 16000
--------------------------------------------------------------------------------
ENTER THE SPECIFIC VOLUME(S) DESIRED BELOW (UP TO 10):
===> ss1802
ALLOCATE THE VOLUMES IN THE ORDER SHOWN? ===>      (YES/NO)
--------------------------------------------------------------------------------

IF MODELING DESIRED, GIVE THE MODEL'S DATASET NAME AND PASSWORD (IF REQUIRED):
MODEL ===>                                                  PASSWORD ==>
IF NNEDED, ENTER THE CATALOG NAME (AND ITS PASSWORD) WHERE MODEL CAN BE FOUND:
CATLG ===>                                                  PASSWORD ==>
DISPLAY MODEL'S ATTRIBUTES ON SUCCEEDING PANELS? ===>        (YES/NO)
```

In this instance, however, an alternate index is being defined over a base cluster. The special panel used for alternate indexes is displayed in Figure G-13.

If the base cluster was defined immediately before the alternate index, the base cluster's name is automatically carried forward to this panel. However, if the alternate index is being defined at a later time, the related cluster's name can be supplied by the user.

FIGURE G-13 Specify Alternate Index Information Panel

```
VAM008 ------------ SPECIFY ALTERNATE INDEX INFORMATION ------------ CA-ALLOCATE
COMMAND ===>

ALTERNATE INDEX NAME:  .OCTOBER.WORKSHOP.KSDS
--------------------------------------------------------------------------------
ENTER THE NAME OF THE BASE CLUSTER WHICH THIS ALTERNATE INDEX IS TO BE
ASSOCIATED WITH:

        RELATED DSNAME ===>
RELATED DSN'S PASSWORD ===>

--------------------------------------------------------------------------------
  PLEASE ANSWER YES OR NO TO THE FOLLOWING QUESTIONS:

 UNIQUEKEYS ===> n     MUST THE ALTERNATE INDEX HAVE UNIQUE KEYS?
    UPGRADE ===> y     UPDATE THE ALTERNATE INDEX WHENEVER THE BASE IS UPDATED?
```

Again, if an exact name is not known, a pattern name can be supplied that presents a scrollable list of names to choose from. The yes or no questions can be answered, or be allowed to default.

This is the abbreviated cluster definition panel.

FIGURE G-14 Abbreviated Cluster Definition Panel (Cluster)

```
VAM0011 --------------- ABBREVIATED CLUSTER DEFINITION -------------- CA-ALLOCATE
COMMAND ===>

CLUSTER NAME ===> july.workshop
CLUSTER TYPE ===> k       (KSDS, ESDS, RRDS, LINEAR, OR AIX)
--------------------------------------------------------------------------------
NUMBER OF RECORDS TO BE ALLOCATED IN DATA SET:
             PRIMARY   ===> 2000
             SECONDARY ===>

RECORD SIZE:    AVERAGE ===> 100        MAXIMUM ===>
    CI SIZE:    DATA    ===>            INDEX   ===>        (KSDS OR AIX ONLY)

            VOLUME NAME ===> ssl803

IF MODELING DESIRED, GIVE THE MODEL'S DATASET NAME AND PASSWORD (IF REQUIRED):
MODEL ===>                                                  PASSWORD ==>

---------------------------- FOR KSDS AND AIX ONLY -----------------------------

KEY INFORMATION: LENGTH ===> 8        OFFSET ===>
 FREE SPACE PCT: PER CI ===>          PER CA ===>
```

As can be seen, there is a minimal amount of information that the user can supply. This simplifies VSAM definition processing for the novice user. It can also be used for those clusters that are used infrequently and do not need the tuning parameters that VSAM affords. If the cluster type specified is an alternate index (AIX), panel VAM008 is displayed after this one to allow the name of the related cluster to be entered. If the model name is entered with a pattern character specified in the name, a catalog locate is issued to find all clusters that match the given pattern.

Those names that match are displayed on a scrollable list so the user can select the data set name that he or she wants to use as a model.

After the definition has taken place, control continues as it does for the full definition process. That is, a panel is displayed showing any messages that IDCAMS may have issued. Also, the user can specify whether the definition commands should be displayed or saved. See Figure G-9, "IDCAMS DEFINE Definition Results Panel," for more details.

Here are the results of the define.

FIGURE G-15 IDCAMS DEFINE Definition Results Panel

```
VAM005 ------------------- DEFINITION RESULTS ----------- CLUSTER DEFINE SUCCESS
COMMAND ===>

CLUSTER NAME:          .JULY.WORKSHOP                          TYPE:KSDS
--------------------------------------------------------------------------------
DISPLAY IDCAMS COMMANDS? ===>       (YES/NO)
   SAVE IDCAMS COMMANDS? ===>       (YES/NO)
   DISPLAY CATALOG INFO? ===>       (YES/NO)
--------------------------------------------------------------------------------
IDCAMS MESSAGES:                                                MSG TOTAL: 4

IDC0508I DATA ALLOCATION STATUS FOR VOLUME SSL803 IS 0
IDC0509I INDEX ALLOCATION STATUS FOR VOLUME SSL803 IS 0
IDC0512I NAME GENERATED-(D)          .JULY.WORKSHOP.DATA
IDC0512I NAME GENERATED-(I)          .JULY.WORKSHOP.INDEX
```

Again, this is the same panel that is displayed when the full cluster definition process is executed.

Here is an example of a define of an alternate index using the abbreviated cluster definition application.

FIGURE G-16 Abbreviated Cluster Definition Panel (AIX)

```
VAM0011 --------------- ABBREVIATED CLUSTER DEFINITION -------------- CA-ALLOCATE
COMMAND ===>

CLUSTER NAME ===> .JULY.WORKSHOP.aix
CLUSTER TYPE ===> A       (KSDS, ESDS, RRDS, LINEAR, OR AIX)
--------------------------------------------------------------------------------
NUMBER OF RECORDS TO BE ALLOCATED IN DATA SET:
             PRIMARY   ===> 50
             SECONDARY ===>

RECORD SIZE:    AVERAGE ===> 20000      MAXIMUM ===> 100
    CI SIZE:    DATA    ===>            INDEX   ===>        (KSDS OR AIX ONLY)

            VOLUME NAME ===> SSL803

IF MODELING DESIRED, GIVE THE MODEL'S DATASET NAME AND PASSWORD (IF REQUIRED):
MODEL ===>                                                  PASSWORD ==>

--------------------------- FOR KSDS AND AIX ONLY ------------------------------

KEY INFORMATION: LENGTH ===> 15       OFFSET ===> 9
 FREE SPACE PCT: PER CI ===>          PER CA ===>
```

Here is the primary panel used for listing catalog information about a cluster.

FIGURE G-17 LISTCAT Utility Primary Panel

```
VAM0015 ---------------------- LISTCAT UTILITY --------------------- CA-ALLOCATE
COMMAND ===>

    CLUSTER NAME  ===> OCTOBER.WORKSHOP.KSDS
CLUSTER PASSWORD  ===>

          CATALOG ===>









    PRESS ENTER TO REQUEST CLUSTER INFORMATION - END TO EXIT LISTCAT
```

If the cluster is read protected, the proper password must be specified. If the cluster is defined in an improper catalog, the catalog name must be entered so that catalog management can find the information for the cluster. If the exact name of the cluster is not known, a pattern name can be entered.

As with the other ISPF functions, BrightStor CA-Allocate issues a locate to find all matching data set names. A scrollable list of these names is then presented for the user to select from.

Here is a sample listing of a cluster's catalog information.

FIGURE G-18 LISTCAT Utility Display Panel

```
VAM0016 -------------------- PRIMARY LISTCAT DISPLAY ------------ Row 1 to 2 of 2
COMMAND ===>__                                                    SCROLL ===> CSR

    CLUSTER:        .OCTOBER.WORKSHOP.KSDS                 TYPE:KSDS
 IN CATALOG: ICFCAT.VSSL801                                           =============
RECORD SIZE: AVE= 1024  MAX= 4089          CI SIZE= 4096             | -SPLITS-  |
 ALLOCATION: DAATA=(CYL,(2,2))      INDEX=(TRK,(1,1))                | CI=     0 |
 FREE SPACE: PCT PER CI=  0           PCT CI'S PER CA=  0             | CA=     0 |
 SPACE INFO: TRKS ALLOC=      31  IDLE=     31  PCT USE=  0           =============
EXTENT INFO: TOTAL EXTS=     2      DATA VOL=SY9SY1   INDEX VOL=SY9SY1

STATS: ADD=         0 DEL=        0 EXCPS=        0 GETS=        0 TOT=     0
DATES: CREATE=08JAN2001     LAST MOD=                   EXPIRES=
 ATTR: IMBED, REPL, SHR(2,3), UNIQUE
 DFLT: NOERASE, NOREUSE, NOSPAN, NOWCK, RECOVERY, UNORDER

PRESS ENTER TO CONTINUE - UP AND DOWN TO SCROLL ASSOCIATION NAMES
================================================================================
CLUSTER ASSOCIATIONS  (TOTAL =   2       )      TYPE
--------------------------------------------   --------
        .OCTOBER.WORKSHOP.KSDS.DATA             DATA
        .OCTOBER.WORKSHOP.KSDS.INDEX            INDEX

******************************** Bottom of data ********************************
```

This screen's layout was patterned after the CLD report used in the VSAM component of BrightStor CA-Disk Backup and Restore. If more than three associations exist for a cluster, the association name list can be scrolled using the up and down keys.

Glossary

ASR - Allocation Selection Routine. A CLIST-like language used to direct BrightStor CA-Allocate processing, or a program written in this language.

CSA Quota Table - The copy of the Quota Table maintained in CSA memory. It is dynamically updated during each data set allocation operation.

DADSM - Direct Access Device Storage Management is the collection of IBM routines that manage access to DASD volumes.

DFHSM - IBM's Data Facility Hierarchical Storage Manager product manages data set and volume backup and recovery. It also manages the archival and recall of data sets.

Disk Quota Table - The copy of the Quota Table maintained on disk. It is synchronized with the CSA Quota Table at the interval you specify in the QSYNCINTERVAL parameter of the OPTIONS statement in the Quota Configuration File.

Environment - A BrightStor CA-Allocate environment denotes the point in the allocation process at which BrightStor CA-Allocate is executing. Different actions may be coded in the VDSPROG ASR, depending on the environment, whose name is given in the &VAMENVIR variable.

Esoteric Unit name - A user defined unit name used to group units according to installation specifications rather than actual device type, defined through an EDTGEN, IOGEN, full SYSGEN, MVSCP, or HCD utility. The only system-generated ESOTERIC unit name is SYSALLDA.

Generic Unit name - A unit name defined by the system for each type of device defined at an installation.

Examples: 3390, 3380, 3350.

IGGPOST0 - An IBM-provided exit point at the end of DADSM.

IGGPRE00 - An IBM-provided exit point at the beginning of DADSM.

LSPACE - A DADSM Macro used to retrieve DASD volume information. It provides free space, volume fragmentation, and VTOC status information.

Nonspecific Volume Request - An allocation request where no volume serial number(s) are specified or referenced.

QREBUILD Operation - The Quota Table rebuild operation. During this operation, the VTOCs of all selected volumes are scanned and allocation statistics are rebuilt. This operation is initiated manually by the QREBUILD operator command. It is initiated automatically whenever the Quota Table is modified in the Quota Configuration File by the QRESET or QUOTADEF statements.

QSYNC Operation - The Quota Table synchronization operation performed by the Started Task. This occurs at the interval you specify in the QSYNCINTERVAL parameter of the OPTIONS statement in the Quota Configuration File.

Quota Configuration File - The member in the PDS referenced by the VDSPROG DD statement (default name QCONFIG) containing statements that define Quota options and Quota Groups.

Quota Table - The data structure where allocation statistics are maintained for the Quota Groups. There are two types of Quota Tables: the Disk Quota Table (DQT—on disk) and the CSA Quota Table (CQT in CSA memory).

Specific Volume Request - An allocation request calling for specific volume serial number(s), or implying them through a reference. In batch this could be through the VOL=SER= parameter on a DD statement, or through the VOL=REF= parameter.

STC - A Started Task that executes on an MVS operating system. As delivered, the default name for the BrightStor CA-Allocate STC is VAM. A sample of the VAM PROC is displayed in FIGURE 3-2 The Cataloged Procedure.

SVC 26 - This is the supervisor call used to invoke a Catalog Management Request.

SVC 32 - This is the supervisor call used to invoke a DADSM allocation function.

SVC 99 - This is the supervisor call used to do a dynamic allocation.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z